కంప్యూటర్ రంగంలోని అద్భుత ప్రోగ్రామింగ్ పుస్తకం

'సి' నేర్చుకుందాం



కె. కిరణ్ కుమార్

పన్నాల వేణుగోపాల్

ముందు మాట

ఒక ప్రోగ్రామింగ్ లాంగ్వేజ్‌ని తెలుగులో రాయడమంటే కత్తి మీద సాము చేయడంలాంటిది. ఇంగ్లీషు పదాలకు తెలుగు పదాలు వెతుక్కోవడం దగ్గర నుండి ఎలా చెబితే చదువరులకు అర్థం అవుతుందో దాకా చాలా శ్రమించాల్సి వుంటుంది.

ఈ పుస్తకం కేవలం 'సి' అంటే ఏమిటో తెలియనివారికే సుమా! ప్రోగ్రామింగ్ అంటేనే భయపడేవారికి Easynessని చేకూర్చడమే ఈ పుస్తకం ప్రథమ లక్ష్యం! అంతేకాని వందలాది ప్రోగ్రామ్‌లు గుప్పించి అవి చేస్తే చాలు 'సి' వచ్చేసిందన్న భ్రమకు గురిచేయడం మా ఉద్దేశం కాదు.

నిజానికీ ప్రోగ్రామింగ్‌కి కావాల్సిందల్లా లాజికల్‌గా ఆలోచించి, ఆ సమస్యల్ని పరిష్కరించడం. అందుకే సమర్ధుడైన ప్రోగ్రామర్ ఎప్పుడూ తాను తెలుసుకున్న బేసిక్స్‌తో కొన్ని సమస్యలను వాస్తవ దృక్పథం నుండి తీసుకొని పరిష్కరించగలగాలి. అంతేకాని రెడీమేడ్ ప్రోగ్రామ్‌లు చేసి 'అమ్మయ్య, నాకు సి వచ్చేసింది' అని తనకు తాను సంతృప్తి పడటం కాదు. ఈ నేపథ్యంలో ఈ పుస్తకంలో మేము బేసిక్స్‌తో పాటు చిన్న ప్రోగ్రామ్‌లు ఎలా రాయాలో చెప్పాం. గురువెప్పుడూ శిష్యులకు అక్షరాభ్యాసం చేస్తారే కాని మంచి కవిత్వం సృష్టించడం మటుకు శిష్యుల సృజనాత్మకత, ఆలోచనా విధానం మీదే ఆధారపడి వుంటుంది కదా!

ఇంకో విషయం! ఎప్పుడైనా ప్రోగ్రామ్‌లు పుస్తకంలో అలా చూస్తుంటే వచ్చినట్లే వుంటాయి. కాని ప్రాక్టికల్‌గా చేస్తుంటే ఎన్నెన్నో తప్పులు కన్పిస్తాయి. కాబట్టి ఏ కంప్యూటర్ సబ్జెక్టయినా నేర్చుకోవాలంటే యాభై శాతం పుస్తకాలు ఉపయోగపడుతుండగా, మిగతా యాభైశాతం కంప్యూటర్ మీద అభ్యాసం తప్పనిసరిగా వుండాల్సిందే!

అంతేకాదు! ప్రస్తుతం ప్రోగ్రామింగ్ లాంగ్వేజ్‌లలో విప్లవం సృష్టిస్తున్న జావా నేర్చుకోవాలంటే ముందు 'సి' లాంగ్వేజ్‌లో కొంత పరిజ్ఞానం తప్పనిసరి. కాబట్టి జావా నేర్చుకొనే విద్యార్థులకు ప్రాథమికంగా కొంత ప్రోగ్రామింగ్ లాంగ్వేజ్‌ను పరిచయం చేయడానికి నాందిగా ఈ పుస్తకాన్ని మేము రాయడం జరిగింది. కాబట్టి ఆ అవకాశాన్ని ఈ పుస్తకం ద్వారా ఔత్సాహికులు సద్వినియోగం చేసుకోవచ్చు.

కాబట్టి ఈ పుస్తకంలో బేసిక్స్ నేర్చుకొని వాస్తవిక దృష్టితో ప్రోగ్రామ్‌లు (ఉదాహరణకు సూపర్ బజార్‌లో వాడే ఓచర్ లేదా ఎలక్ట్రిక్‌సిటీ బిల్లింగ్ ప్రోగ్రామింగ్ వెనకాల జరిగే లాజిక్) తెలుసుకొని కంప్యూటర్ ద్వారా ప్రాక్టీస్ చేస్తే ఉజ్వలమైన భవిష్యత్తు వుంటుంది. విష్ యూ ఆల్ ది బెస్ట్!

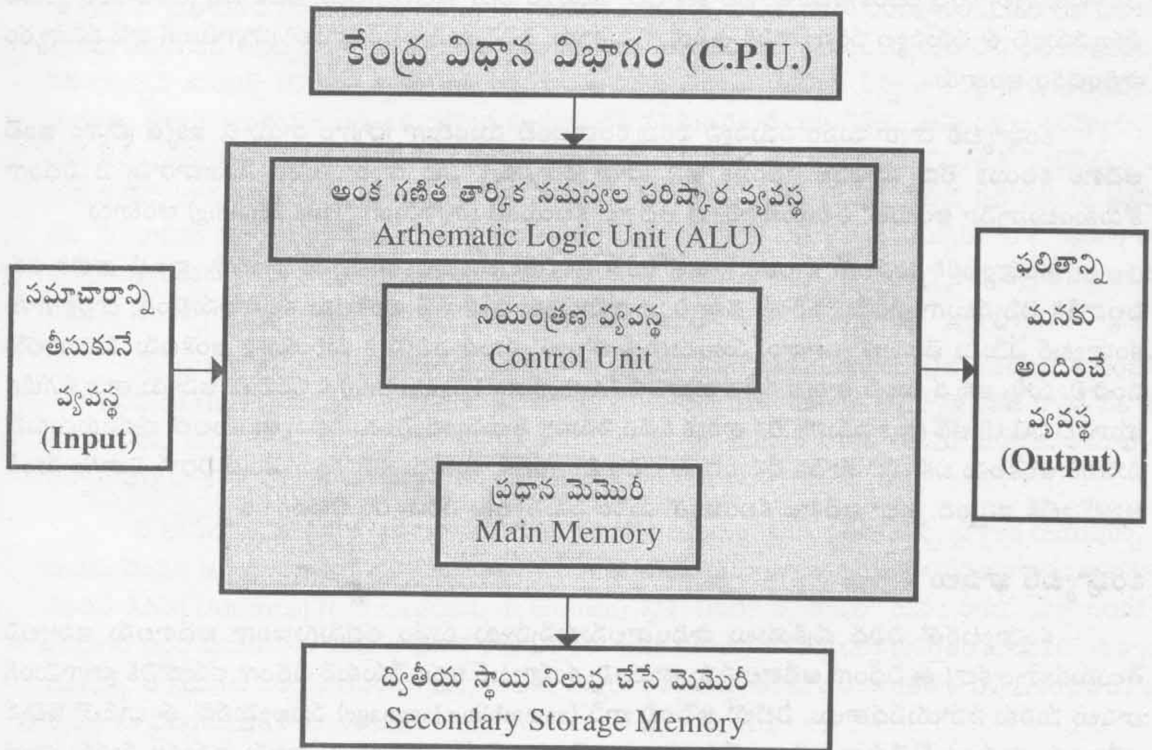
- రచయితలు

విషయ సూచిక

అధ్యాయం 1

కంప్యూటర్ భాషలు

కంప్యూటర్ అంటే సమాచారాన్ని తనలోకి తీసుకొని (Input) వాటిని కొన్ని భాగాలుగా విడగొట్టి విశ్లేషించి (Processing), తిరిగి వాటి తాలూకు ఫలితాన్ని (Output) మనకు చూపించే పరికరం. కంప్యూటర్‌లో వివిధ రకాల క్రియలు ఈ క్రింది విధంగా జరుగుతాయి.



పై వాటిలో చాలా ముఖ్యమైనది సెంట్రల్ ప్రాసెసింగ్ యూనిట్ (CPU). ఇక్కడే కంప్యూటర్ ఇన్‌పుట్ ద్వారా మనం పంపించిన సమాచారాన్ని భద్రపరుస్తుంది. అయితే కంప్యూటర్ మనం పంపిన సమాచారాన్ని అలాగే తీసుకోదు. దానిని తన భాష అయిన అసెంబ్లీ భాషలోకి మార్చుకుంటుంది. ఎందుకంటే కంప్యూటర్ మన భాషని అర్థం చేసుకోలేదు కాబట్టి.

ప్రోగ్రామింగ్ - ప్రాథమిక పరిజ్ఞానం

పైన తెలిపిన కంప్యూటర్ ద్వారా మనం ముఖ్యంగా రెండు రకాల పనులను సాధించగలం. అవి-

1. సమాచారాన్ని ఇచ్చి పుచ్చుకోవడం

2. సమస్యలకు పరిష్కారాలను పొందడం.

పై పనులలో ఒకటవదైన సమాచారాన్ని ఇచ్చి పుచ్చుకోవడానికి మనకు సాంకేతిక పరిజ్ఞానం, తార్కిక జ్ఞానం అవసరం లేదు. కేవలం కొన్ని రకాల ప్యాకేజీలు నేర్చుకుంటే చాలు. ఇక్కడ ప్యాకేజీలు అంటే ప్రోగ్రామ్‌లు రాయనవసరం లేకుండా కేవలం కొన్ని బొమ్మలు (Icons) ను గుర్తుపెట్టుకొని పనిచేసుకోవడానికి ఉపయోగపడే సాఫ్ట్‌వేర్‌లు. ఉదాహరణకు యం.ఎస్. ఆఫీసు, ఫాక్స్ ప్రో లాంటివి ప్యాకేజీలు. ఈ ప్యాకేజీల ఆధారంగా మనం ప్రోగ్రామింగ్ పరిజ్ఞానం లేకుండానే మన ఆఫీసుకు లేదా వ్యాపారానికి కావాల్సిన డాక్యుమెంట్లు, టేబుళ్ళు, స్టేట్‌మెంట్ల తాలుకూ లెక్కలు, లేఖా చిత్రాలు తయారు చేసుకోవచ్చు.

ఇక కంప్యూటర్ యొక్క రెండవ ముఖ్య ఉద్దేశం ఏమిటంటే - **సమస్యలకు పరిష్కారాలను పొందడం**. క్లిష్టమైన ఎన్నో సమస్యల్ని కంప్యూటర్‌కి నివేదిస్తే, అది క్షణాలలో పరిష్కారాలను కనుగొనగలదు. నిజం చెప్పాలంటే కంప్యూటర్‌ని ఎక్కువమంది ఈ సమస్యల పరిష్కారానికే ఉపయోగిస్తున్నారు. దీనినే కంప్యూటర్ భాషలో ప్రోగ్రామింగ్ రాసి పరిష్కారం సాధించడం అంటారు.

కంప్యూటర్ ద్వారా మనం సమస్యని పరిష్కరించాలంటే ముందుగా ప్రోగ్రాం రాయాలి. **ఇక్కడ ప్రోగ్రాం అంటే ఆదేశాల కలయిక లేదా సూచనల కలయిక అని కూడా చెప్పవచ్చు**. వీటి ద్వారా మనం సమాచారాన్ని ఏ విధంగా కోరుకుంటున్నామో అందులో తెలియజేస్తాం. ఆ ఆదేశాల కలయికనే ప్రోగ్రామింగ్ (Programming) అంటారు.

కంప్యూటర్లకి మనలాగే స్వంతంగా ఆలోచించే తెలివితేటలు, విచక్షణా జ్ఞానం వుండవు. కాబట్టి వాటికి చిన్న పిల్లవాడికి చెప్పినట్లుగా ప్రతిదీ వివరించి చెప్పాలి. ఉదాహరణకు 2+2 అనే అంకెలను కలపాలనుకోండి. దానికి గాను కంప్యూటర్ ఏమేమి చేయాలో చూద్దాం. ముందుగా కీ బోర్డులో నుండి వచ్చిన 2 మరియు 2 అంకెలను మెమొరీలోకి పంపాలి. మళ్ళీ అక్కడ నుండి తార్కిక గణిత యూనిట్ (Arthematic Logical Unit) కి పంపాలి. అప్పుడు తార్కిక గణిత యూనిట్ (ALU) అది అంక సమస్యో లేక తార్కిక గణిత సమస్యో తెలుస్తుంది. రెండు విభిన్న అడ్రసులలో భద్రపరుచుకున్న మెమొరీ అంకెలను ఒక చోట కూడిక చేస్తుంది. ఆ ఫలితాన్ని ఇంతకు ముందు చెప్పుకున్న కేంద్ర విధాన విభాగం నుండి అవుట్‌పుట్‌కి ఇస్తుంది. అలా ఆదేశాల కలయికతో మనం సమస్యలకు పరిష్కారం పొందగలం.

కంప్యూటర్ భాషలు

కంప్యూటర్లతో వివిధ ప్రక్రియలు సాధించాలనుకున్నప్పుడు మనం తదనుగుణంగా ఆదేశాలను ఇవ్వాలని తెలుసుకున్నాం కదా! ఈ విధంగా ఆదేశాల సముదాయాన్ని కంప్యూటర్ అర్థం చేసుకునే విధంగా ఇవ్వడానికి ప్రోగ్రామింగ్ భాషలు మనకు సహాయపడతాయి. వీటిలో అసెంబ్లీ భాష (assembling language) ముఖ్యమైనది. ఈ భాషలో ఇచ్చిన ఆదేశాలను కంప్యూటర్ నేరుగా అమలు చేస్తుంది. అయితే ఈ అసెంబ్లీ భాషలో ఆదేశాలను ఇవ్వడం మనకు చాలా కష్టమైన పని.

ఒక చిన్న ప్రక్రియను జరపాలన్నా అందుకు చాలా ఆదేశాలను ఇవ్వాల్సి ఉంటుంది. అలాగే చిన్న ప్రక్రియకు ప్రోగ్రాం రాయాలన్నా అది చాలా పెద్ద ప్రోగ్రామవుతుంది. పైగా మన వాడుక భాషకు దూరంగా ఉంటుంది కనుక అసెంబ్లీ భాషలో ఆదేశాలను గుర్తుంచుకోవటం కూడా కష్టమే. అంతేకాకుండా ఒకరకం కంప్యూటర్ల కోసం రాసిన అసెంబ్లీ భాష మరో రకం కంప్యూటర్లకు పనికి రాదు కూడా.

అసెంబ్లీ భాషతో ఉన్న ఇబ్బందులను తొలగించడానికి మనకు హైలెవల్ ప్రోగ్రామింగ్ భాషలు ఉపయోగపడతాయి. ఈ హైలెవల్ భాషలలో ఆదేశాలు ఇంగ్లీషు భాషకు దగ్గరగా ఉంటాయి. అంతేకాక ఏ రకం కంప్యూటర్‌పై అయినా ఒకే విధమైన ఆదేశాలను ఉపయోగించవచ్చు. అయితే అన్ని రకాల కంప్యూటర్లు ఒకే విధమైన ఆదేశాలను ఎలా అర్థం చేసుకుంటాయన్న అనుమానం వచ్చింది కదూ! మనం హైలెవల్ భాషలో రాసిన ఆదేశాలు తిరిగి అసెంబ్లీ భాషలోకి తర్జుమా అవుతాయి. నిజానికి కంప్యూటర్ అర్థం చేసుకోగలిగేది అసెంబ్లీ భాషనే. హైలెవల్ భాషలో మన ఒక్కో ఆదేశం అనేక అసెంబ్లీ ఆదేశాలుగా మారతాయి. **ఈ విధంగా హైలెవెల్ భాషలోని ఆదేశాలను అర్థం చేసుకుని వాటిని అసెంబ్లీ భాషలోని ఆదేశాలుగా మార్చే సాధనాన్నే కంపైలర్** (Compiler) **అంటారు.** పాస్కల్, కోబాల్, సి, ఫోర్ట్రాన్ వంటివి హై లెవెల్ భాషలకు ఉదాహరణలు. ఏ హైలెవెల్ భాషకు సంబంధించిన కంపైలర్ దానికే విడిగా ఉంటుంది. హైలెవెల్ భాషలోని ఆదేశాలన్నింటినీ కంపైలర్ దానంతటదే అసెంబ్లీ భాషలోకి మారుస్తుంది. ఈవిధంగా అసెంబ్లీ భాషలోకి మారిన ఆదేశాలను కంప్యూటర్ అమలు చేస్తుంది. మనకు అందుబాటులో ఉన్న హైలెవెల్ భాషలలో అత్యంత శక్తివంతమైన 'సి' భాష. దీని గురించి ఇప్పుడు చదువుదాం.

సి-భాష చరిత్ర

గడచిన రెండు దశాబ్దాల కాలంలో 'సి' భాషకు ప్రపంచ వ్యాప్తంగా బ్రహ్మాండమైన ప్రాచుర్యం లభించింది. సాఫ్ట్‌వేర్ పరిశ్రమ స్వరూపాన్నే సి భాష మార్చివేసిందంటే అది అతిశయోక్తి కాదు. సాఫ్ట్‌వేర్ రంగంలో ఎన్నో కొత్త భాషలు వస్తున్నా 'సి' ప్రభావం ఏ మాత్రం తగ్గలేదు. ఇంతటి ప్రాచుర్యం ఉన్న 'సి' భాషను 1970 సంవత్సరంలో బెల్ లేబొరేటరీస్‌కు చెందిన డెన్నిస్ రిచీ అభివృద్ధి చేశారు. 'సి' భాష అవతరణ కొంత ఆసక్తికరంగానే ఉంటుంది. బెల్ లేబొరేటరీస్‌కు చెందిన కెన్ థాంప్సన్ యునిక్స్ (UNIX) ఆపరేటింగ్ సిస్టమ్‌ను అభివృద్ధి చేశారు. థాంప్సన్ దీన్ని అసెంబ్లీ భాష ఉపయోగించి తయారు చేశారు. అందువల్ల ఇది ఒక రకమైన కంప్యూటర్‌కే పరిమితమైంది. యునిక్స్‌ను అన్ని రకాల కంప్యూటర్లపైన చిన్న చిన్న మార్పులతో ఉపయోగించేలా తయారు చేయ్యాలని బెల్ నిర్ణయించింది. ఇందుకోసం ఒక హైలెవెల్ భాషను ఉపయోగిస్తే బాగుంటుందని భావించారు. అప్పటికే అందుబాటులో ఉన్న 'బి' భాషను డెన్నిస్ రిచీ మరింత అభివృద్ధి చేసి 'సి' భాషను రూపొందించారు. ఇది అన్ని విధాలుగా అనుకూలంగా ఉండటంతో యునిక్స్‌ను 'సి'లో తిరగరాశారు. ప్రస్తుతం వాడుకలో ఉన్న యునిక్స్‌లో 90 శాతం 'సి' భాష ఉపయోగించి రాసిందే. ఈ విధంగా యునిక్స్ కోసం అభివృద్ధి పరచిన 'సి' ఇప్పుడు బహుళ ప్రచారం పొంది కంప్యూటర్ రంగాన్ని ఒక వూపు వూపేస్తోంది.

'సి' భాషకు ఆపేరు ఎలా వచ్చిందో చూద్దాం. ఈ భాషను కనుగొనకముందు బెల్ లేబొరేటరీస్‌లోని ఇంజినీర్లు 'బి', 'బిసిపిఎల్'(BCPL) అనే భాషలను తయారు చేశారు. డెన్నిస్ రిచీ వీటిని మరింత అభివృద్ధి పరచి దానికి ఏం పేరు పెట్టాలా అని ఆలోచించి 'సి' అని నామకరణం చేశారు. నిజానికి 'సి' అనేపేరు యాదృచ్ఛికమే. ఇంగ్లీషు అక్షరాల్లో 'B' తర్వాత వచ్చేది 'C' కాబట్టి ఈ భాషకు ఆ పేరు లభించింది.

సి భాషలో వివిధ రకాల ఆదేశాలు ఉన్నాయి. ఈ ఆదేశాలను ఒక క్రమపద్ధతిలో ఇచ్చి ప్రోగ్రాం రాయవచ్చు. 'సి' భాషకు చెందిన ఆదేశాలకు, వాటిని రాసే పద్ధతికి కొన్ని అంతర్జాతీయ ప్రమాణాలున్నాయి. ఈ ప్రమాణాలను అమెరికాకు చెందిన ANSI (American National Standards Institute) నిర్దేశించింది. 'సి' భాషలో మనం రాసిన ప్రోగ్రాంలను 'సి' కంపైలర్ అర్థం చేసుకుని అసెంబ్లీ భాషలోకి మారుస్తుంది. అనేక రకాల 'సి' కంపైలర్లు ప్రస్తుతం వాడుకలో ఉన్నాయి. యునిక్స్, డాస్ వంటి వేర్వేరు ఆపరేటింగ్ సిస్టమ్‌లకు వేర్వేరు కంపైలర్లు ఉంటాయి. అంతేకాక వివిధ కంపెనీలు తమ స్వంత కంపైలర్లను తయారు చేశాయి.

ఈ పుస్తకంలో TURBO C కంపైలర్‌ను ఉపయోగించి ప్రోగ్రామ్‌లు రాశాం. నిజానికి చిన్న తేడాలు మినహా అన్ని కంపైలర్లకి మధ్య పెద్ద భేదం లేదు. యునిక్స్‌లో కూడా ప్రోగ్రాం రాసే విధానం ఇలాగే ఉంటుంది. అయితే, ఆపరేటింగ్ సిస్టమ్‌కు సంబంధించి చిన్న చిన్న తేడాలుంటాయి. అవసరమైన చోట్ల ఈ పుస్తకంలో ఆ తేడాలను వివరించాం. ఇక 'సి' భాషను నేర్చుకోవడం ఎలాగో తర్వాతి పాఠం నుంచి చూద్దాం!

✦✦✦

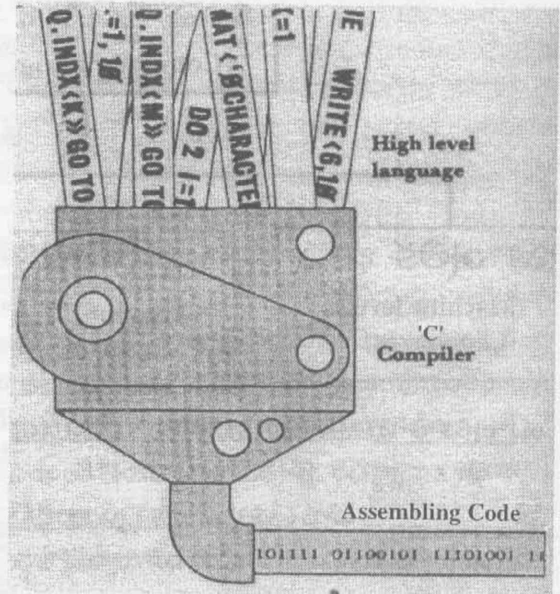
అధ్యాయం 2

సి కంపైలర్-ఉపయోగించే తీరు

సి భాషను కంప్యూటర్‌కి అర్థమయ్యే భాషలోకి మార్చాలంటే కంపైలర్ వుండాలని ఇంతకు ముందు చదివారా! ఇప్పుడు ఆ కంపైలర్ అంటే ఏమిటి ? దానికై ఎలా పనిచేయాలి ? అన్న అంశాలు ఈ పాఠంలో చూద్దాం.

కంప్యూటర్‌లు కేవలం ఎలక్ట్రానిక్ యంత్రాల్లాంటివి. దురదృష్టవశాత్తూ కంప్యూటర్ మనం మాట్లాడే భాషలైన తెలుగు, ఇంగ్లీషు, హిందీ లాంటివి అర్థం చేసుకోలేవు. అవి కేవలం తమ భాషైన అసెంబ్లీ భాషనే అర్థం చేసుకోగలవు.

ఫలితంగా మన భాషని కంప్యూటర్ల తమ భాషైన అసెంబ్లీ భాషలోకి మార్చుకోక తప్పదు. ఈ నేపథ్యంలో మనం, పంపిన భాషని కంప్యూటర్ భాషలోకి తర్జుమా చేయడానికి 'కంపైలర్' (Compiler) అనే సాధనం వచ్చింది. అంటే ఇక్కడ ఈ కంపైలర్ ఓ దుబాసీలా పనిచేస్తుందన్న మాట. ఉదాహరణకు కంప్యూటర్ నిపుణులు తరచూ వాడే 'సి' భాషని కంప్యూటర్ భాషలోకి మార్చాలంటే 'C' కంపైలర్ కావాలి. ఇక్కడ ఇవ్వబడిన బొమ్మలో కంపైలర్ మన 'C' భాషని తన అసెంబ్లీ భాషలోకి ఎలా మార్చుకుంటుందో చూడండి.



నిజానికి ఒక్కో భాషకు ఒక్కో కంపైలర్ వుంటుంది. ఈ కంపైలర్ ఆపరేటింగ్ సిస్టంను బట్టి కూడా మారుతూ వుంటుంది. ఉదాహరణకు 'సి' భాషకు ఒక కంపైలర్ వుంది. అదే డాస్, యూనిక్స్, వంటి వివిధ ఆపరేటింగ్ సిస్టమ్‌లలో 'సి' భాషకు వేర్వేరు కంపైలర్లు ఉన్నాయి. ఈ నేపథ్యంలో మన 'సి' భాషకు తరుచూ వాడే TURBO C కంపైలర్ గురించి ఇక్కడ తెలుసుకుందాం.

TURBO C కంపైలర్

'సి' భాషని రాయడానికి ప్రోగ్రామర్లు తరుచుగావాడే కంపైలర్ నే 'టర్బో సి' కంపైలర్ అంటారు. 'సి' ప్రోగ్రాం రాయటానికి ఇది ఒక ఎడిటర్‌ను కూడా సమకూరుస్తుంది. (సాధారణంగా TURBO C++ కంపైలర్‌ను కూడా వాడుతూంటారు. ఇది కూడా సి భాషను కంపైల్ చేయడానికి ఉపయోగిస్తుంది.) ఇప్పుడు 'సి' కంపైలర్ అయిన TURBO C కంపైలర్‌ని మీ కంప్యూటర్‌లో ఎలా లోడ్ చెయ్యాలో చూద్దాం. అందుకోసం TURBO C.EXE అనే ఫైల్‌ని మీ కంప్యూటర్‌లోకి లోడ్ చెయ్యాలి. ఆ తర్వాత డాస్‌లో

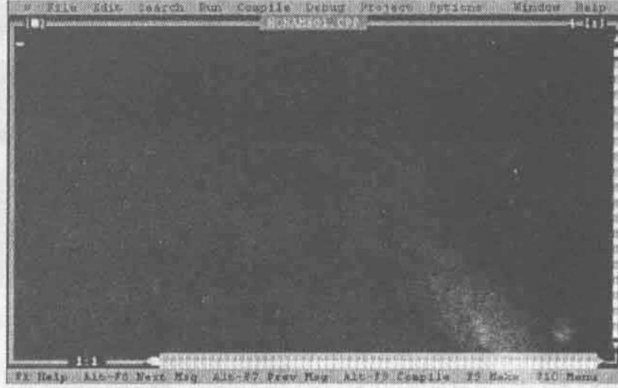
C:\\ TURBO C అన్న ఆదేశం ఇవ్వాలి.

అప్పుడు 'సి' లేదా 'సి++' కంపైలర్ మీ కంప్యూటర్‌లో ఇన్‌స్టాల్ అవుతుంది.

అప్పుడు తిరిగి డాస్ ప్రాంప్ట్‌లో

C:\>TC

అన్న ఆదేశం ఇస్తే మీ ముందు ఈ కింది తెర ప్రత్యక్షమవుతుంది.



దీన్నే TURBO C ఎడిటర్ అంటారు.

ఈ ఎడిటర్‌లో పైన మెనూలు ఉన్నాయి. అందులో FILE అని ఉన్న మెనూను తెరవాలి. ఏదైనా మెనూను ఓపెన్ చేయాలంటే Alt.కీని నొక్కి ఆ మెనూ పేరులోని మొదటి అక్షరం ఉన్న కీని నొక్కాలి. ఉదాహరణకు FILE మెనూను చూడాలనుకుంటే Alt.కీని నొక్కి పట్టుకుని F కీని నొక్కాలి. అప్పుడు మెనూ ఓపెన్ అవుతుంది. బాణం గుర్తులున్న కీల సహాయంతో మనకు కావలసిన ఐటంను సెలెక్ట్ చేసుకోవచ్చు. ఒక కొత్త ప్రోగ్రాం రాయలంటే FILE మెనూలోని New ను సెలెక్ట్ చేసుకోవాలి. అప్పుడు నీలంగా ఉన్న మరో తెర ప్రత్యక్షమవుతుంది. దానిపై ఒక కర్సర్ వస్తుంది. ఆ తెరపై మనం ప్రోగ్రాంను టైప్ చేయవచ్చు.

```
≡ File  Edit  Search  Run  Compile  Debug  Project  Options    Window  Help
=[■]=========================== PROG2.C ===========================1=[↕]=
main()
{
      /* Examples for ASCII characters */
      int num1,num2;
      num1 = 65;
      /* ASCII code for A */
      num2 = 97;
      /* ASCII code for a */
      /* Now let us print the characters */
      pintf("The characters are %c %c \n",num1,num2);
}
```

ఈ ప్రోగ్రాంను కంపైల్ చేయాలంటే Compile మెనూ (Alt.+C) లో Compile ను సెలక్ట్ చేయాలి లేదా Alt.+F9 ను నొక్కాలి. ఇప్పుడు ప్రోగ్రాంలో ఏవైనా తప్పులు ఉంటే ఈ కింద చూపిన విధంగా ఎర్రర్ కనిపిస్తుంది.

```
≡  File  Edit  Search  Run  Compile  Debug  Project  Options    Window  Help
#include<stdio.h>
main()
{
        /* Examples for ASCII characters */
        int num1,num2;
        num1 = 65;
        /* ASCII code for A */
        num2 = 97
        /* ASCII code for a */
        /* Now let us print the characters */

}

Compiling PROG2.C:
•Error PROG2.C 11: Statement missing ;
Warning PROG2.C 12: Function should return a value
Warning PROG2.C 12: 'num2' is assigned a value that is never used
Warning PROG2.C 12: 'num1' is assigned a value that is never used
F1 Help  Space View source  ← Edit source  F10 Menu
```

తప్పు (Error) జరిగినచోటు ←

దాన్ని సరిచేసేక మళ్ళీ కంపైల్ చేస్తే ప్రోగ్రాం కంపైల్ అవుతుంది. ఈ ఎర్రర్‌లలో రెండు రకాలున్నాయి. మొదటిది Warning ప్రోగ్రాంలో చిన్న తప్పులుంటే ఇది హెచ్చరిస్తుంది. దీని వల్ల సాధారణంగా ప్రోగ్రాంలో ఎటువంటి తప్పులూ రావు. అయితే అవి కూడా లేకుండా జాగ్రత్త పడటం మంచిది. కాని మొదట్లో ఈ warning ని పట్టించుకోవాల్సిన అవసరం లేదు. ఇక error అని వచ్చిన వాటిని సరిచెయ్యాలి. లేకపోతే ప్రోగ్రాం పని చెయ్యదు. ఏ లైన్‌లో ఎర్రర్ వచ్చిందో, అది ఎందుకు వచ్చిందో చూసి దాన్ని సరి చెయ్యాలి. 'సి' ప్రోగ్రామింగ్ లాంగ్వేజ్‌లో వున్న గొప్పతనమేమిటంటే మనం ప్రోగ్రామ్‌ని తప్పుగా టైప్ చేసినా, మనకు అది తెలయజేస్తుంది. దాంతో మనం ఆయా తప్పులను సరిచేసుకోవచ్చు.

ప్రోగ్రాం ఎటువంటి తప్పులూ లేకుండా కంపైల్ అయితే దాన్ని రన్ చేయాలి. రన్ చేయాలంటే Run మెనూలో Run ను సెలెక్ట్ చెయ్యాలి. అప్పుడు ప్రోగ్రాం రన్ అవుతుంది. అయితే ఆ రన్ అయిన ప్రోగ్రాం మనకు కనిపించదు. అది డాస్ తెరపై రన్ అవుతుంది. దాన్ని చూడాలంటే window మెనూలో user screen ను సెలక్ట్ చేయాలి. అప్పుడు డాస్ తెరపై మన ప్రోగ్రాం ఫలితం కనిపిస్తుంది. ఈ మెనూలను సెలెక్ట్ చేయటం కొంత పెద్ద పనిగా ఉంది కదూ. అందుకనే షార్ట్‌కట్‌లను ఉపయోగించుకోవచ్చు. ఈ షార్ట్ కట్‌లను ప్రక్క టేబుల్‌లో ఇచ్చాం పరిశీలించండి.

షార్ట్‌కట్	ఉపయోగం
Alt.+F3	మెమొరీలో ఉన్న ఫైల్‌ను ఓపెన్ చేయటానికి
F3	ప్రస్తుతం ఉన్న ప్రోగ్రాంను క్లోన్ చేయటానికి
Alt.+X	TURBO Cని క్లోజ్ చేయటానికి
F9	కంపైల్ చేయడానికి
Ctrl.+F9	కంపైల్ చేసి రన్ చేయడానికి
Alt.+F5	డాస్ తెరను ఓపెన్ చేయడానికి
F1	TURBO C లో Help కోసం

ఏదైనా అనుమానం వుంటే F1 వస్తే హెల్ప్‌ను చూడండి. ఎర్రర్ మీకు అర్థం కాకపోయినా లేదా ఏదైనా పదాన్ని ఎలా ఉపయోగించాలో తెలీకపోయినా ఆ పదంపై కర్సర్‌ను ఉంచి Ctrl.+F1 నొక్కితే Help ప్రత్యక్షమవుతుంది. తద్వారా మీ సందేహాలు తీర్చుకోవచ్చు.

✦ ✦ ✦

అధ్యాయం 3

సి ప్రోగ్రాం-స్వరూపం
(C Programme–Nature)

మనుషులు తమ మనసులోనే భావాలను మరొకరికి తెలియజేయడానికి వాడేదే భాష. మనం ఎవరితో సంభాషిస్తున్నామో వారికి అర్థమయ్యే భాషలో మాట్లాడడానికి ప్రయత్నిస్తాం. పశుపక్ష్యాదులకు ఆదేశాలను ఇవ్వాలనుకున్నప్పుడు వాటికి అర్థమయ్యే భాషలోనే ఇవ్వాలి. అదే విధంగా కంప్యూటర్‌కి ఆదేశాలు ఇవ్వాలనుకుంటే వాటి భాషలోనే ఇవ్వాలి. కాబట్టి దేనికైనా 'భాష' ముఖ్యం.

ప్రతి భాషకు కొన్ని నియమాలు, నిబంధనలు ఉంటాయి. ఏదైనా వాక్యం భాషకు అనుగుణంగా లేని పక్షంలో 'తప్పు' వచ్చిందంటాము. ఉదాహరణకు

నేర్చుకునే సులువైన కష్టపడినవారికి భాష 'సి' చాలా.

పై వాక్యం తెలుగులోనే ఉంది. అయితే ఈ వాక్యం మీకు అర్థం కాలేదు కదూ ? దీన్ని క్రమం మార్చి రాస్తే ఈ కింది విధంగా ఉంటుంది.

కష్టపడి నేర్చుకునే వారికి 'సి' చాలా సులువైన భాష.

అంటే కర్త, కర్మ, క్రియ వంటివి తెలుగుభాష నిర్వచించిన నిబంధనల ప్రకారమే ఉండాలి. ఇదే విధంగా 'సి' భాష రాయడానికి కూడా కొన్ని నియమ నిబంధనలు (Syntax rules) ఉన్నాయి. వాటి ప్రకారమే భాషను రాయాల్సి ఉంటుంది. ఈ నిబంధనలను పూర్తిగా నేర్చుకుంటే 'సి' రాయడం వచ్చేసినట్లే.

ఇప్పుడు 'సి'లో అసలు ప్రోగ్రాం స్వరూపం ఎలా ఉంటుందో చూద్దాం.

ఈ కింది ప్రోగ్రాం చూసి కంగారు పడకండి. ప్రోగ్రాం ఎలా ఉంటుందో చూపడానికి మాత్రమే దీన్ని ఇస్తున్నాం. మీకేకాదు మొదటిసారి చూసేవారికి ఎవరికైనా ఇది ఖచ్చితంగా అర్థంకాదు.

```
# include<stdio.h>  →  హెడర్ ఫైల్
main ()             →  మెయిన్
{                   →  ప్రారంభ బ్రాకెట్
    int num =1, count, i;                       ┐
    long result;                                │
    printf ("Enter number to find factorial") ; │
    scanf ("%d", & num);                        ├── ప్రోగ్రాం
    for (i=num-1; i>=1; i--)                    │
    result * = num-1;                           │
    printf ("Factorial is %ld", result);        ┘
}         →  ముగింపు బ్రాకెట్.
```

హెడర్ ఫైల్ : దీని గురించి వివరంగా మరో అధ్యాయంలో చర్చించుకుందాం. ప్రస్తుతానికి ప్రతి ప్రోగ్రాంలోను ఇటువంటి హెడర్‌ఫైల్‌ను తప్పనిసరిగా రాయాల్సి ఉంటుందని మాత్రం తెలుసుకుంటే సరిపోతుంది. ఇందులో మనం ఉపయోగించే STDIO అంటే Standard Input Output అని అర్థం.

మెయిన్

ప్రతి ప్రోగ్రాంకి ఇది తలకాయలాంటిది. ప్రతి 'సి' ప్రోగ్రాంను తప్పనిసరిగా దీనితోనే ప్రారంభించాలి. main అని రాసిన తర్వాత () బ్రాకెట్లు కూడా ఇవ్వాలి.

దీని తర్వాత అసలైన ప్రోగ్రాం ప్రారంభమవుతుంది. అయితే 'సి' లో ప్రోగ్రాం మొత్తం { } బ్రాకెట్ (curly braces) మధ్యే ఉండాలి. అంటే { బ్రాకెట్ తర్వాత ప్రోగ్రాం ప్రారంభం కావాలి. అంతా పూర్తయ్యాక } బ్రాకెట్ ఉండాలి.

ఇప్పుడు మనం 'సి'లో మొదటి ప్రోగ్రాం రాద్దాం. డాస్‌లో లేదా విండోస్‌లో అయితే ముందుగా TCC డైరెక్టరీలో ఉన్న TC. EXE ని ఎగ్జిక్యూట్ చేస్తే మీ స్క్రీన్‌పై 'సి' ప్రోగ్రామింగ్ విండో ప్రత్యక్షమవుతుంది. అందులో ఈ కింది ప్రోగ్రాంను టైప్ చేయండి.

```
# include< stdio.h>
main()
{
    printf("Hello world\n");
}
```

ఇప్పుడు File మెనూలోని save ఆప్షన్‌ను సెలక్ట్ చేసి ఈ ఫైలును **Hello.C** గా సేవ్ చేయండి. ప్రతి 'సి' ప్రోగ్రాంను **.c** ఎక్స్‌టెన్షన్‌తో సేవ్ చేయాలి. సాధారణంగా TC (Tourbo C) ఎన్విరాన్‌మెంట్‌లో సేవ్ చేస్తే అది Hello.c గానే సేవ్ అవుతుంది. ఇప్పుడు కీ బోర్డుపై Ctrl కీని, F9 కీని ఒకేసారి నొక్కండి. అప్పుడు ఆ ప్రోగ్రాం ముందు కంపైల్ అవుతుంది. తర్వాత రన్ అవుతుంది. ఫలితంగా మీ స్క్రీన్‌పై

Hello World

అని output ప్రత్యక్షమవుతుంది.

స్క్రీన్‌పైన మీ Output
చూడాలంటే alt+F5 నొక్కండి.

ఇప్పుడు పై ప్రోగ్రాంలోని భాగాలను పరిశీలిద్దాం. ఇది 'సి'లో అతి చిన్న ప్రోగ్రాం. ఇందులో ముందుగా మనం #include<stdio.h> అన్న హెడర్ రాశాము. తర్వాత main() రాశాము. తర్వాత { బ్రాకెట్‌తో ప్రోగ్రాం ప్రారంభమయింది.

మనకు ఇందులో ఒక కొత్త అంశం కనిపించింది. అది printf అనే ఫంక్షన్. స్క్రీన్‌పై ఏదైనా ప్రింట్ చేయాలనుకుంటే దీన్ని ఉపయోగిస్తాం. పై ఉదాహరణలో మనం Hello world అనే పదాలను ప్రింట్ చేయాలనుకున్నాం. ఆ పదాలను printf (" ") ఫంక్షన్‌లో రాశాము. printf అని రాసిన తర్వాత '(' బ్రాకెట్ ప్రారంభమవుతుంది. 'సి'లో ఏ ఫంక్షన్ (Function) అయినా '(' బ్రాకెట్‌తో ప్రారంభమయి ')' బ్రాకెట్‌తో ముగుస్తుంది. అందుకనే main అనే ఫంక్షన్ పక్కన కూడా () బ్రాకెట్లు ఉన్నాయి గమనించారా! అయితే ఈ ఫంక్షన్‌లో మనం ఏమీ రాయలేదు. అదే printf అనే ఫంక్షన్ లోపల మనం

Hello world అని రాశాం. కంప్యూటర్ తెరపై ఏది కనిపించాలో దాన్ని మనం printf ఫంక్షన్‌లో రెండు కొటేషన్ మార్కుల మధ్య (" ") రాస్తాము. నిజానికి ఈ కొటేషన్ మార్కుల మధ్య రాసినదే తెరమీద కనిపిస్తుంది.

నిజానికి రెండు కొటేషన్ మార్కులు కనిపించినా అది ఒకటే గుర్తు కీబోర్డుపై ఉన్న double quotes ను ఉపయోగించి వీటిని ప్రింట్ చేయవచ్చు. ఈ డబుల్ కోట్స్ ప్రోగ్రాం ఎగ్జిక్యూట్ అయినప్పుడు మనకు కనిపించవు. కేవలం వాటి మధ్య ఉన్న పదాలే ప్రింటవుతాయి. ఏ పదాలను ప్రింట్ చేయాలో కంప్యూటర్‌కు తెలిపేందుకు మాత్రమే ఈ డబుల్ కోట్స్ ఉపయోగపడతాయి. రెండోవైపు డబుల్ కోట్స్ రాగానే ప్రింటింగ్ ఆగిపోతుంది. అయితే ఈ డబుల్ కోట్స్‌ను ప్రింట్ చేసే అవకాశం కూడా ఉంది. ఉదాహరణకు

Hello "WORLD"

అని ప్రింట్ కావాలంటే

printf("Hello"WORLD"");

అని రాస్తే Error (తప్పు) వస్తుంది. ఎందుకంటే ఏ డబుల్ కోట్స్‌ను ప్రింట్‌చేయాలో, ఏ డబుల్ కోట్స్‌ను ఆదేశంగా అర్థం చేసుకోవాలో కంపైలర్ నిర్ణయించుకోలేక అయోమయంలో పడుతుంది. పై పదాలను ప్రింట్ చేయాలంటే printf స్టేట్‌మెంట్ ఈ కింది విధంగా ఉండాలి.

printf ("Hello\"WORLD\"");

ఇప్పుడు Hello "WORLD" అని ప్రింటవుతుంది. \గుర్తు తర్వాత ఉన్న డబుల్ కోట్స్‌ను యధావిధంగా ప్రింట్ చేయాలని కంపైలర్ అర్థం చేసుకుంటుందా? print f స్టేట్‌మెంట్‌లో ఎన్ని పదాలను ప్రింట్ చేయాలన్నా వాటన్నిటినీ రెండు డబుల్‌కోట్స్ (" ") మధ్యే ఉంచాలి. అంటే

Hello! How are you ? అని ప్రింట్ చేయాలంటే

printf ("Hello! How are you?");

అని రాయాల్సిందే!

printf ("Hello!""How""are""you?");

అని రాస్తే ఎర్రర్ వస్తుంది.

ఇక్కడ మనకు '\n' అని మరో రెండు గుర్తులు కనిపిస్తున్నాయి. నిజానికి 'సి' భాషలో ఈ రెండిటినీ ఒకటే గుర్తు character) గా పరిగణిస్తాం. \n ను 'సి'లో న్యూలైన్ క్యారెక్టర్ (newline character) అంటారు. దీని ఉపయోగమేమిటంటే ని తర్వాత ఉండే అక్షరాలు లేదా పదాలు తెరపై మరోలైన్‌లో ప్రింటవుతాయి. అంటే మనం

printf ("Hello World \n Welcome to C");

అని రాస్తే అది తెరపై

Hello World

Welcome to C

అని ప్రత్యక్షమవుతుంది.

ఇక్కడ మరో విషయం గమనించండి. \n అనే క్యారెక్టర్ తెరపై ప్రింట్ కాదు. అంటే కొత్త లైన్ ప్రారంభానికి ఇది మనం ఇచ్చే ఆదేశం మాత్రమే. కొటేషన్ మార్కుల మధ్య మాత్రమే ఈ ఆదేశాన్ని రాయాలి.

మరో ప్రధాన అంశం ఏమిటంటే printf అనే లైన్ చివర సెమికోలన్ (;) గుర్తు ఉంది. అంటే printfతో మొదలైన వాక్యం అక్కడికి పూర్తయిందన్నమాట.

మామూలుగా, ఇంగ్లీషు, తెలుగు వంటి భాషల్లో వాక్యం ముగిసిందనడానికి గుర్తుగా చుక్క (.) ను వాడతాం. అదేవిధంగా 'సి' భాషలో వాక్యం ముగింపుకు గుర్తుకు సెమికోలన్ (;) ను వాడతాం. 'సి'లో వాక్యాన్ని స్టేట్‌మెంట్ (statement) అంటారు. ప్రతి స్టేట్‌మెంట్ తప్పనిసరిగా సెమికోలన్‌తోనే అంతం కావాలి.

'సి' భాషలో మరో విశేషముంది. \n అన్న న్యూలైన్ క్యారెక్టర్ ఉంటే తప్పించి తెరపై రెండోలైన్ ప్రారంభం కాదు. ఉదాహరణకి

```
printf("HelloWorld");
printf("Welcome to C");
```

అన్న స్టేట్‌మెంట్‌లు ఒకే ప్రోగ్రాంలో ఒకదానికింద మరొకటి ఇచ్చినప్పటికీ మనకు తెరపై

HelloWorldWelcome to C.

అనే ప్రింటవుతుంది. కనీసం రెండు వాక్యాల మధ్య కొంత ఖాళీకూడా లేదు. ఎందుకంటే మొదటి వాక్యం చివరగానీ, రెండో వాక్యం మొదట్లోగానీ మనం ఖాళీని ఇవ్వలేదు. దీన్ని బట్టి 'సి'లో భాష మనం ఎలా ఇస్తే తెరపై అలానే వస్తుందని అర్థమవుతోంది కదు. 'సి'లో ప్రత్యేక లక్షణం ఇదే. ప్రోగ్రాంపై ప్రోగ్రామర్‌కు **పూర్తి నియంత్రణ** ఉంటుంది.

ఇక ఇక్కడ ఉన్న మరో ప్రత్యేకత చూద్దాం. పై ప్రోగ్రాంలో మనం main() అని రాసి తర్వాత లైన్‌లో { అని మొదలు పెట్టాం. నిజానికి { బ్రాకెట్ main() పక్కన లేదా main() తర్వాత ఎక్కడైనా ఉండవచ్చు. అదేవిధంగా } బ్రాకెట్‌కూడా ప్రోగ్రాంలో చివరి స్టేట్‌మెంట్ తర్వాత ఎక్కడైనా ఉండవచ్చు. అంటే పై ప్రోగ్రాంను

```
#include<stdio.h>
main(){printf("Hello world");}
```

అని కూడా రాయవచ్చు.

ఈ పుస్తకంలో మాత్రం main() కిందే { బ్రాకెట్ రాసే సంప్రదాయాన్ని పాటించాము.

ఇప్పుడు మరో ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main()
{
    printf("C is easy if you practice \n difficult if you just notice");
}
```

ఈ ప్రోగ్రాంను easy.c గా సేవ్ చేయండి. తర్వాత కీబోర్డుపై ఉన్న Ctrl+F9 సాయంతో కంపైల్ చేసి రన్ చేయండి. మీకు తెరపై ఈ కింది విధంగా కనిపిస్తుంది.

```
C is easy if you practice
difficult if you just notice
```

పై ప్రోగ్రాంను ఈ కింది విధంగా రాస్తే ప్రోగ్రాంను మళ్లీ చదివేవారికి బాగా అర్థమవుతుంది.

```
#include<stdio.h>
main()
{
    printf("C is easy if you practice \n");
    printf("difficult if you just notice");
}
```

'సి' భాషలోకాని మరే ఇతర భాషలోనైనా ప్రోగ్రాములు రాసేటప్పుడు కొన్ని జాగ్రత్తలు తీసుకోవాలి. ముఖ్యంగా ఆ ప్రోగ్రాంను మీరే మళ్లీ చదివినా లేదా మరొకరు చదివినా ఆ ప్రోగ్రాం ఎందుకు రాశారో వెంటనే అర్థమయ్యేట్లుండాలి. మీరు ప్రోగ్రాం రాసిన తర్వాత దాన్ని వేరేవారికి చూపితే, ఆ ప్రోగ్రాం ఉద్దేశమేమిటో వారు సులువుగా చెప్పగలిగి ఉండాలి. అంటే ప్రోగ్రాంను సాధ్యమైనంత సరళంగా రాయాలి. ఉదాహరణకు రెండు లైన్లుగా ప్రింట్ అయ్యే వాక్యాన్ని ఒకే printf ఉపయోగించి ఒకే లైన్‌లో రాస్తే చూసేవారికి వెంటనే అర్థం కాకపోవచ్చు. అదే రెండు లైన్లుగా రాస్తే వెంటనే అర్థమవుతుంది. ఆ ప్రోగ్రాం output ఎలా ఉంటుందో కూడా చూస్తూనే చెప్పవచ్చు.

ఇప్పుడు మరొక విధమైన ప్రోగ్రాంను చూద్దాం.

```
#include<stdio.h>
main()
{
    printf(" C is the                \n");
    printf("            most powerful  \n);
    printf("            language\n");
    printf("            of all the             \n");
    printf("computer languages         ");
}
```

దీన్ని మీరు పైన చెప్పిన విధంగా ఎగ్జిక్యూట్ చేస్తే, తెరపై ఈ కింది విధంగా కనబడుతుంది.

```
C is the
                most powerful
                language
                of all the
computer languages
```

తమాషాగా ఉంది కదూ. printf లోపల ఖాళీలను కీబర్డుపై గల స్పేస్‌బార్ (spacebar) ను ఉపయోగించి సృష్టించవచ్చు.

కంప్యూటర్ ప్రోగ్రాం రాసేటప్పుడు దృష్టిలో ఉంచుకోవాల్సిన మరోముఖ్యమైన అంశం కామెంట్స్ (comments). పైన పేర్కొన్నట్లుగా మనం రాసే ప్రోగ్రాం మరొకరు సులువుగా అర్థం చేసుకునేట్లుండాలి. అంటే ప్రోగ్రాంలోని ప్రతి అంశాన్ని సులువుగా అర్థం చేసుకునేట్లుండాలి. అంతేకాక ప్రోగ్రాంలో ఒక స్టేట్‌మెంట్‌ను ఎందుకు రాశామో కూడా చెప్తే మరింత అర్థవంతంగా ఉంటుంది. ఇందుకు ఉపయోగించేవే **కామెంట్స్**. ప్రోగ్రాం మధ్యలో కామెంట్స్‌ను కింద చూసినట్లుగా రాయవచ్చు.

```
#include<stdio.h>
main( )
{
        /* it is a new program.*/
        printf("Hello, welcome to C\n");
        /* New line character
        is used to print the
        next contents in seperate line */
}
```

ఈ ప్రోగ్రాంలో /* అన్న గుర్తుల తర్వాత సాధారణ ఇంగ్లీషు భాషలో ఒక వాక్యం రాశాం. తర్వాత */ అని రాశాం. ఈ రెండు గుర్తులు - /*, */ మధ్యలో రాసినదాన్ని కామెంట్ అంటారు. దీన్ని కంపైలర్ గుర్తించదు. అంటే /* ...... */ గుర్తుల మధ్యలో ఏమి రాసినా, కంపైలర్ దృష్టిలో మాత్రం అక్కడేమీ లేనట్లే లెక్క. ఈ కామెంట్‌ను ఒక లైన్‌లో మొదలుపెట్టి రెండు మూడు లైన్లు కొనసాగించవచ్చు కూడా. ప్రోగ్రాం రాసిన తర్వాత దాన్ని మరొకరు చదివితే స్పష్టంగా అర్థం కావటానికి ఈ కామెంట్స్ బాగా ఉపయోగపడతాయి. ఎక్కడైనా కొత్త ప్రయోగాలు చేస్తున్నప్పుడల్లా అక్కడ ఏం చేస్తున్నామో తెలియజేయడానికి కామెంట్స్ మనకు బాగా అందుబాటులోకి వస్తాయి. నిజానికి కొత్త అంతర్జాతీయ ప్రమాణాల ప్రకారం ప్రోగ్రాంలో రాసే ప్రతి స్టేట్‌మెంట్‌ను ఎందుకు రాశామో, దాని ఉపయోగం, ఆవశ్యకతలను తెలియజేస్తూ కామెంట్స్‌ను రాయాల్సి ఉంది. అంటే ప్రోగ్రాం కంటే కామెంట్స్ ఎక్కువ ఉండాల్సి వస్తుందేమో. ఈ కింది ఉదాహరణలను గమనించండి.

```
#include<stdio.h>
main ()
{
        /*print publication name */
        printf ("Sai Shiva \n");
        /* print door number */
        printf ("10-68\n");
        /*print colony name*/
        printf ("Sowbhagyanagar\n");
        /*print city and pincode*/
        printf ("HYDERABAD-500 037\n");
        /*End of program*/
}
```

పై ప్రోగ్రాంలో కామెంట్స్ రాయకుండా కంటే మనం అసలు ఏమి రాస్తున్నామన్న సంగతి ప్రోగ్రాం చదివేవారికి అర్థం కాదు. ప్రతి లైన్‌కు కామెంట్ రాసినందువల్ల ప్రోగ్రాంలో ప్రతి స్టేట్‌మెంట్ ఎందుకు రాస్తున్నామో అవతలివారికి స్పష్టమైపోతుంది. మీరు ఏమీ చెప్పకపోయినా, ప్రోగ్రాం తన గురించి తానే చెప్పుకుంటుంది. ఇదే మరో ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main()
{
        /*begin the program*/
        /*print the country*/
        printf ("Native of India\n");
        /*print the state*/
        printf ("Belongs to Andhra Pradesh\n");
        /*print city*/
        printf ("Resident of Hyderabad\n");
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి చూడండి. మీకు printf లో రాసినవి మాత్రమే కనిపిస్తాయి. ప్రోగ్రాం చదివే వారికి మాత్రమే కామెంట్స్ ఉపయోగపడతాయి.

అభ్యాసం

ఈ కింది OUTPUTS వచ్చే విధంగా ఒక 'సి' ప్రోగ్రాం రాయండి.

1. **Good Morning. How are you ?**

2. **Johny Johny! What papa ?**
Eating sugar ? No papa !
Telling lies ? No papa !
Open your mouth ! Ha Ha Ha!
(సూచన : ప్రతి లైన్ ప్రింట్ చేయడానికి printf స్టేట్‌మెంట్‌లను వాడవచ్చు)

3.
```
                 Please
         Check          this
      program             properly
   so that                    you write
      correct             program
         without        any
                 Errors
```



అధ్యాయం 4

చరరాశులు-వాటి ప్రకటన

(Integers-their Declaration)

కంప్యూటర్‌లో ఎంతో సమాచారాన్ని భద్రపరచుకోవచ్చని, అందుకోసం మెమొరీని ఉపయోగిస్తామని మీకు తెలుసు. ప్రోగ్రామ్ రాసేటప్పుడు ఆ మెమొరీలో కొన్ని విలువలను నిలవ చేస్తాం. మళ్లీ అవసరమైనప్పుడు వాటిని మెమొరీలోంచి బయటకు తీసుకుని వాడుకుంటాం. అయితే ఈ విధంగా నిలవ చేయడానికి, వెనక్కు తీసుకోడానికి అసలు మెమొరీలో విలువను ఎక్కడ నిలవ ఉంచామనే విషయం మనకు తెలియాలి. ఉదాహరణకి 3 అనే అంకెను మెమొరీలో ఒక ప్రదేశంలో భద్రపరిచామనుకుందాం. దాన్ని మళ్లీ వెనక్కు తీసుకోవాలనుకుంటే అది ఎక్కడ ఉందో తెలియాలి. దానికోసం ఆ అంకెను భద్రపరిచిన ప్రదేశానికి మనం ఒక పేరు పెడతాం. ఇప్పుడు 3 అనే అంకె భద్రపరిచిన ప్రదేశానికి మనం 'x' అని పేరు పెడదాం. అంటే x లో ఉన్న విలువ 3 అన్నమాట. మరో విధంగా చెప్పాలంటే x విలువ 3 అని అర్థం.

మెమొరీలో ఒక ప్రదేశానికి x అని పేరు పెట్టాం. అందులో 3ను నిలవ ఉంచితే x విలువ 3 అవుతుంది. అదే 3కు బదులుగా 4ను నిలవ ఉంచితే x విలువ 4 అవుతుంది. ఈ విధంగా x విలువను మనం మార్చవచ్చు. విలువను మార్చే వీలున్నందున x ను చరరాశి (variable)అంటారు. అదే 3 లేదా 4 అన్న విలువలను మనం మార్చలేము. కాబట్టి వాటిని స్థిరరాశులు (constants) అంటారు. మనం పాఠశాలలో చదివిన ఆల్జీబ్రాలో ఈ స్థిరరాశుల గురించి చదివాం కదా! ఇక్కడ కూడా అదేవిధంగా స్థిరరాశులను ఉపయోగించవచ్చు.

'సి' భాషలోకాని లేదా ఏ ఇతర కంప్యూటర్ భాషలోనైనాసరే చరరాశులను వివిధ విభాగాలుగా లేదా రకాలుగా విభజించవచ్చని, కంప్యూటర్ మెమొరీలో విలువలను నిలవ చేయవచ్చని తెలుసుకున్నాం కదా! అయితే ఎటువంటి విలువలను మనం నిలువ చేయవచ్చు? 'సి' భాషలోని అన్ని రకాల అక్షరాలు, అంకెలు, గుర్తులను మనం నిలవ చేయవచ్చు.

గమనిక: ప్రతీ భాషలోని లిపికి కొన్ని అక్షరాలు, గుర్తులు, అంకెలు ఉంటాయి. ఉదాహరణకు ఇంగ్లీషు భాషలో a, b, c, d ...., A, B, C ...., 1, 2, ... వంటి అక్షరాలు, అంకెలు ఉన్నాయి. అదే 'అ, ఆ, ఇ, ఈ .... వంటి అక్షరాలను ఇంగ్లీషు భాషలో రాయలేము. అదే విధంగా ప్రతి భాషకు కొన్ని అక్షరాలు, గుర్తుల సముదాయం (Character set) ఉంటుంది. 'సి' భాషకు కూడ అదే విధమైన గుర్తుల సముదాయం ఉంది. అందులో ఈ కింది అక్షరాలు, గుర్తులు ఉన్నాయి.

అక్షరాలు : A నుంచి Z వరకు,

a నుంచి z వరకు

అంకెలు : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

గుర్తులు : ~ ' ! @ # % ^ & * ( ) _ - + = \ | { } [ ] : ; " ' . . ? /

వీటిని మాత్రమే 'సి' భాష మెమొరీలో నిలవ చేయగలరు. అయితే వీటి అన్నిటికీనీ ఒకేవిధమైన చరరాశిని ఉపయోగించి నిలవచేయటం సాధ్యం కాదు. ఎందుకంటే అక్షరాలు, అంకెలు ఒకేరకమైన మెమొరీని తీసుకోవు. ఒక్కో అక్షరం లేదా గుర్తుకి ఒక్కో బైట్ మెమొరీ కావాలి. అదే విధంగా సంఖ్యలకు వాటి పరిమాణాన్ని బట్టి రెండు నుంచి ఎనిమిది బైట్‌ల మెమొరీ కావాలి. అక్షరాలను లేదా గుర్తులను నిలవ చేయటానికి char రకం చరరాశులను, అంకెలను నిలవ చేయటానికి int (integer) రకం చరరాశిని ఉపయోగిస్తాం.

సంఖ్యలలో కూడా అనేక రకాలున్నాయి. సాధారణ సంఖ్యలను int రకం చరరాశిని ఉపయోగించి మెమొరీలో భద్రపరచవచ్చు. అదే దశాంశస్థానం లేదా ఘనస్థానాలు (Decimal or exponents) ఉన్నప్పుడు float రకం చరరాశిని ఉపయోగించాలి. ఉదాహరణకు

1.325, 12^{10} వంటి విలువలను నిలవ ఉంచడానికి float రకం చరరాశిని ఉపయోగించవచ్చు.

ఈ విలువలను ఉపయోగించి ఇప్పుడు ఒక ప్రోగ్రాం రాద్దాం.

```
#include<stdio.h>
main()
{
        printf("Print the number %d",26);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

Print the number 26

అని తెరపై ప్రింటవుతుంది.

పై ప్రోగ్రాంలో కొన్ని ప్రత్యేక అంశాలున్నాయి. ముఖ్యంగా %d అనే గుర్తులను ఉపయోగించాము. 26 అనే సంఖ్యను కొటేషన్ గుర్తుల బయట రాశాం. నిజానికి %d అనేది ఒకటే గుర్తు. దీన్ని Conversion specification అంటారు. integer విలువలను తెరపై ప్రింట్ చేయడానికి దీన్ని వాడతారు. ఒక్కోరకం విలువకు ఈ కన్వర్షన్ స్పెసిఫికేషన్ ఒక్కో విధంగా ఉంటుంది.

ఉదాహరణకు

```
printf("%d", 43.497);
```

అని రాస్తే 43 మాత్రమే ప్రింటవుతుంది. అంటే విలువలోని పూర్ణభాగం (integral part) మాత్రమే ప్రింటవుతుంది. దశాంశ స్థానంతో సహా ప్రింట్ చేయాలంటే float రకం వాడాలి. దీనికి %f ను కన్వర్షన్ స్పెసిఫికేషన్‌గా ఉపయోగిస్తాం.

```
printf("%f", 43.497);
```

అనిరాస్తే అప్పుడు 43.497 ప్రింటవుతుంది.

ఇప్పటి వరకూ మనం printf లో ఇచ్చిన సంఖ్యలను ప్రింట్ చేసే విధానం చూశాం. మెమొరీలో విలువలను భద్రపరచే విధానాన్ని ఇప్పుడు చూద్దాం. int రకం విలువలను నిలవ చేయడానికి int చరరాశులు, float సంఖ్యలను నిలవచేయడానికి float చరరాశులను, అదే అక్షరాలు, గుర్తులను నిలవ చేయడానికి char చరరాశులను ఉపయోగిస్తాం.

*మనం ఏదైనా చరరాశిని ప్రోగ్రాంలో ఉపయోగించదలచుకుంటే అది ఏరకమైన చరరాశి అన్న విషయాన్ని ప్రోగ్రాం ప్రారంభంలో తప్పనిసరిగా తెలియజేయాలి. దీన్నే చరరాశుల ప్రకటన (**Variable Declaration**) అంటారు. మనం ప్రోగ్రాం రాయడానికి ఏరకమైన విలువలను తీసుకున్నామో, వాటిని కంప్యూటర్‌కి తెలియజేయడానికి అలా చేస్తారు.*

ఒక ప్రోగ్రాంలో ఉపయోగించే చరరాశులన్నిటినీ, ఆ ప్రోగ్రాం ప్రారంభంలో ప్రకటించాలి. ప్రోగ్రాంకు సంబంధించిన మరే వాక్యమైనా చరరాశుల ప్రకటన తర్వాతే ఉండాలి.

చరరాశులని ప్రకటించాలంటే ముందుగా అవి ఏరకానికి చెందినవో రాసి తర్వాత చరరాశిని రాయాలి. ఉదాహరణకు x అనే చరరాశి int రకానికి చెందినది అనుకుందాం. ఇప్పుడు x ని ప్రకటించాలంటే

```
int x;
```

అని రాయాలి. ముందు int అనిరాసి తర్వాత ఒక ఖాళీ (space) ఇచ్చి అప్పుడు చరరాశిని రాయాలి. చివరన సెమికోలన్ ఉండాలి. రెండు లేదా అంతకన్నా ఎక్కువ చరరాశులను కూడా ఒకే లైన్‌లో ప్రకటించవచ్చు. ఉదాహరణకి x, y, z అనే మూడు చరరాశులని int చరరాశులుగా ప్రకటించాలంటే

```
int x, y, z;
```

అని రాయాలి. చరరాశుల మధ్య కామా (,) ఉండాలి. చివరి చరరాశి తర్వాత సెమికోలన్ ఉండాలి. చరరాశుల మధ్య ఖాళీ ఉండాలన్న నిబంధన లేదుకానీ కొంత ఖాళీ (space) ఇస్తే చదవటానికి వీలుగా ఉంటుంది.

ఇదే float రకానికి చెందిన చరరాశిని ప్రకటించాలంటే

```
float x;
```

అని రాయాలి. char అదే రకానికి చెందిన చరరాశిని కూడా

```
char x;
```

అని ప్రకటించవచ్చు.

చరరాశులను ఎలా ఉపయోగించాలో ఈ కింది ప్రోగ్రాంలో చూద్దాం.

```
#include<stdio.h>
main()
{
    int x, y, z;
    float a;
    char p, q;
    x = 3;
    y = 67;
    z = 15;
    a = 43.75;
```

```
    p = 's';
    q = 't';
    printf("%d, %d, %d, %f, %c, %c", x,y,z,a,p,q);
}
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి రన్ చేస్తే,

3, 67, 15, 43.75, s, t

అని ప్రింటవుతుంది.

char చరరాశులని ప్రింట్ చేయడానికి %c అన్న conversion factor ను ఉపయోగించాలి. x కి విలువ ఇవ్వాలంటే

x = 3; అని రాశాం.

అంటే x కి 3 అని విలువను ఇచ్చాం. దీన్ని assign చేయటం లేదా assignment అంటారు.

ఈ assignment ప్రక్రియలో చరరాశి ఎప్పుడూ = గుర్తుకి ఎడమవైపునే ఉండాలి.

ఇప్పుడు x విలువ 3 కదా ఇప్పుడు

y = x;

అనిరాస్తే, x లో ఉన్న విలువ y లోకి కూడా చేరుతుంది. అంటే y విలువ కూడా 3 అవుతుంది.

char రకం చరరాశులు p, q లకు s, t అనే అక్షరాలను ఇచ్చాం. అంటే p లో s ను, v లో t ని భద్రపరిచాం. ఈ assignment లో s, t లను కోట్స్‌లో (single quotes) ఇచ్చాం గమనించండి. ఇక్కడ s, t అనేవి చరరాశులు కావు. స్థిరరాశులు. ఆ విషయం చెప్పటానికే వాటిని కోట్స్‌లో ఉంచాము.

ఇప్పుడు మరో ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main()
{
    int x, y;
    x = 2;
    y = 4;
    printf("%d", x+y);
}
```

పై ప్రోగ్రాంలో మనం x కు 2, y కి 4 విలువలను ఇచ్చాం. ఇప్పుడు ఈ రెంటి మొత్తాన్ని కనుగొని తెరపై ప్రింట్ చేయమని చెప్పాం. తెరపై మీకు 6 కనిపిస్తుంది.

ఈ చరరాశులను మీరు ఏవిధంగానైనా తీసుకోవచ్చు. ఇవి కేవలం x, y లేదా a, b అనే ఉండాలన్న నిబంధన లేదు. అర్థవంతమైన పదాలను కూడా వాడుకోవచ్చు. ఉదాహరణకు

number, digit, value వంటి పదాలను ఉపయోగించవచ్చు.

చరరాశులు అర్థవంతంగా ఉంటే చూసేవారికి ప్రోగ్రాం సరిగా అర్థమవుతుంది. ఈ కింది ప్రోగ్రాంలో చరరాశులను పరిశీలించండి.

```
#include<stdio.h>
main()
{
    int number_1, number_2, sum;
    number_1 = 10;
    number_2 = 15;
    sum = number_1+number_2;
    printf("%d", sum);
}
```

పై ప్రోగ్రాం రన్ చేస్తే తెరపై 25 ప్రింటవుతుంది. ఈ ప్రోగ్రాంని చూసే వారికి మనం ఏం చేయనున్నామో ప్రత్యేకంగా చెప్పక్కరలేదు. రెండు చరరాశులకి 10, 15 అనే విలువలను ఇస్తున్నాము. వాటిని కూడి, ఫలితాన్ని మూడో చరరాశి sum లో నిలువ చేస్తున్నాం. తర్వాత printf ను ఉపయోగించి ఆ sum ను ప్రింట్ చేస్తున్నాం.

పై ప్రోగ్రాంను మరింత వివరంగా పరిశీలిద్దాం.

number_1 అనే చరరాశికి 10 అనే విలువను ఇచ్చాం. ఏ చరరాశి తర్వాతయినా Is Equal to (=) గుర్తును ఇచ్చి తర్వాత ఒక విలువను ఇస్తే, ఆ విలువ ఆ చరరాశికి చేరుతుంది.

x=1, అంటే x అనే చరరాశికి 1 అనే విలువ ఇస్తున్నామని అర్థం. = గుర్తుకి కుడిచేతి పక్కన ఏదో ఒక సంఖ్య ఉండాలన్న నిబంధనేదీ లేదు. మరో చరరాశి కూడా కావచ్చు.

y = x అని కూడా రాయవచ్చు. అప్పుడు y అనే చరరాశిలో 1 అనే విలువ చేరుతుంది. x లోని విలువ మారదు. అంటే ఈ విలువను ఇచ్చే Assignment లో కుడిచేతివైపున ఉన్న చరరాశి లేదా స్థిరరాశుల విలువ మారదు. కేవలం ఎడం చేతి వైపున ఉన్న వాటి విలువలు మాత్రమే మారతాయి. అందువల్ల మనం

4 = 20 అని రాయటం తప్పు. ఎందుకంటే 4 విలువ మారదు.

అదేవిధంగా 4 = x అని కూడా రాయకూడదు.

కూడికలు, తీసివేతలు వంటివి కుడిచేతివైపున చేయవచ్చు. అంటే

x = 4+7 అని రాయవచ్చు. అప్పుడు x లో విలువ 11 అవుతుంది.

అదేవిధంగా x = 2;

y = 4;

z = x+y అని రాయవచ్చు. అయితే ఈ విధంగా రాసేటప్పుడు x, y, z లు ఒకే విధమైన సమాచారాన్ని (same data type) కలిగి ఉండాలి.

చరరాశుల పేర్లకు ఇంగ్లీషు పదాలను ఉపయోగించవచ్చనుకున్నాం కదా! అయితే ఈ విధంగా ఉపయోగించడానికి కొన్ని నిబంధనలున్నాయి. చరరాశులకు పేర్లను పెట్టేటప్పుడు ఈ కింది నిబంధనలను తప్పనిసరిగా అనుసరించాలి.

1. ప్రతి చరరాశి పేరు తప్పనిసరిగా a–z మధ్యగల ఒక ఇంగ్లీషు అక్షరంతోనే ప్రారంభం కావాలి. అండర్‌స్కోర్ (_) ను కూడా చరరాశిపేరులో ఉపయోగించవచ్చు. నిజానికి అండర్‌స్కోర్‌ను పేరు మొదట్లో కూడా ఉపయోగించవచ్చుకాని ప్రస్తుతానికి అలా వాడవద్దు.

2. పేరులో రెండవ అక్షరం నుంచి ఇంగ్లీషు అక్షరాలు (a నుంచి z వరకు) లేదా అంకెలు (0, 1, 2, 3, ...) లేదా అండర్‌స్కోర్‌ను ఉపయోగించవచ్చు.

3. చరరాశి పేరు పొడవుపై కూడా పరిమితి ఉంది. పేరు పొడవు 8 అక్షరాలను మించకూడదు. (కొన్ని కంప్యూటర్లు 32 అక్షరాలను అనుమతిస్తాయి). అంటే మొత్తం పేరులో అక్షరాలు, అంకెలు, అండర్‌స్కోర్ కలిపి 8ని మించకూడదు.

మనం ఒకవేళ 8 అక్షరాలను మించి ఇచ్చినా 'సి' కేవలం మొదటి 8 అక్షరాలనే పరిగణనలోకి తీసుకుంటుంది. ఉదాహరణకి మనం రెండు చరరాశులకు number_12, number_13 అని పేర్లు పెట్టామనుకోండి. ఇందులో number_1 వరకే ఎనిమిది అక్షరాలు పూర్తయ్యాయి. కాబట్టి 'సి'కి రెండు పేర్లు ఒకే విధంగా కనిపిస్తాయి. అందువల్ల ఎర్రర్ (తప్పు) ఇచ్చే అవకాశం ఉంది.

4. ఇంగ్లీషు భాషలో పెద్దపడి (Upper case), చిన్నపడి (Lower case) అక్షరాలున్నాయి. పెద్దపడిలో అక్షరాలను A, B, C, D...... అని రాస్తాం. చిన్నపడిలో a, b, c, d..... అని రాస్తాం. (మామూలుగా ఈ రెండింటికి పెద్ద తేడా ఉండదు.) 'సి' భాష ఈ రెంటిని వేర్వేరుగా పరిగణిస్తుంది. సి భాషకు చెందిన పదాలు (ఉదాహరణకు main, include) కేవలం చిన్నపడిలోనే ఉంటాయి. అంటే చిన్నపడి అక్షరాలనే ఉపయోగించాలి. అయితే చరరాశుల పేర్లలో పెద్దపడి (Upper case) అక్షరాలను ఉపయోగించవచ్చు. పెద్దపడి, చిన్నపడి అక్షరాలను జాగ్రత్తగా ఉపయోగించాలి. ఉదాహరణకు Number, numBer లను సి భాష వేర్వేరు చరరాశులుగా పరిగణిస్తుంది. సాధారణంగా చరరాశుల పేర్లకు కూడా చిన్న పడి అక్షరాలను ఉపయోగించటం ఆనవాయితీ.

5. 'సి' భాషకు చెందిన పదాలను చరరాశుల పేర్లుగా ఉపయోగించకూడదు. పై ప్రోగ్రాంలను పరిశీలించండి. int, include, main వంటి పదాలను మనం ప్రతి ప్రోగ్రాంలోనూ ఉపయోగించాం. అదేవిధంగా సి భాష తనకంటూ ప్రత్యేకంగా కొన్ని పదాలను రిజర్వు చేసుకుంది. అందువల్ల వాటిని ప్రధాన పదాలు (Key words) లేదా రిజర్వు పదాలు (Reserved words) అంటారు. సి భాషలో 32 రిజర్వు పదాలున్నాయి. అవి

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

(కొన్ని కంపెనీలు తయారు చేసిన 'సి' కంపైలర్లు మరికొన్ని పదాలను కూడా రిజర్వు పదాలుగా గుర్తిస్తాయి. ఆ కంపైలర్లలో Help చూస్తే వాటి గురించి తెలుస్తుంది.)

6. అండర్‌స్కోర్ (_) తప్ప మిగిలిన గుర్తులు (కామా, చుక్క, సెమికోలన్ వంటివి) చరరాశి పేరులో ఉండకూడదు.

చరరాశులు కొంత విలువను నిలవ ఉంచుకుంటాయని తెలుసు. printf స్టేట్‌మెంట్‌లో %d అనే గుర్తు ఈ చరరాశి విలువను ప్రింట్ చేస్తుంది. ఒకే printf స్టేట్‌మెంట్‌లో ఒకటికన్నా ఎక్కువ %d గుర్తులుండవచ్చు. కాని ఎన్ని చరరాశుల విలువలను ప్రింట్ చేయదలచుకున్నామో అన్ని %d లను మాత్రమే ఇవ్వాలి. ఉదాహరణకు ఈ కింది స్టేట్‌మెంట్‌ను గమనించండి.

```
printf("There are %d days, %d weeks and %d months in a year", d, w, m);
```

చరరాశులను కొటేషన్ తర్వాత కామా ఇచ్చి రాయాలి. ప్రతి చరరాశికి మధ్య కామా ఉండాలి. చివరి చరరాశి తర్వాత బ్రాకెట్ ')' ఉండాలి. పై స్టేట్‌మెంట్ d, w, m అనేవి చరరాశులు. వాటిని ప్రింట్ చేయడానికి కొటేషన్ మార్కుల లోపల మూడు %d గుర్తులున్నాయి. ఎడమనుంచి కుడివైపుకి మొదటి %d గుర్తు చరరాశి d లో విలువని, రెండో %d గుర్తు చరరాశి w విలువను, మూడో %d గుర్తు చరరాశి m విలువని ప్రింట్ చేస్తాయి.

ఒకవేళ మనం d = 365, w = 52, m = 12 అన్న విలువలను ఇస్తే అప్పుడు printf స్టేట్‌మెంట్ ఫలితం ఈ కింది విధంగా ఉంటుంది.

There are 365 days, 52 weeks and 12 months in a year. ఇదే మనం d, w, m, లను కొటేషన్ గుర్తుల లోపల %d గుర్తు లేకుండా ఇస్తే

There are days, weeks and months in a year అని ప్రింటవుతుంది.

అభ్యాసం

1. a = 2

   b = 3

   c = a+ b

   పై ఈ సమీకరణానికి ప్రోగ్రాం రాసి c విలువను కనుగొనండి

2. a = 10

   b = 20

   c = 30

   d = 40

   e = a + b c + d దీనికి ప్రోగ్రాం రాసి c విలువను కనుగొనండి.

✦ ✦ ✦

అధ్యాయం 5



'సి'లో గణిత ప్రక్రియలు (Arithmetic Operations in 'C')

గణిత శాస్త్రంలో మనకు నాలుగు మౌలిక ప్రక్రియలున్నాయి. అవి కూడిక, తీసివేత, హెచ్చవేత, భాగహారం. ఈ నాలుగు ప్రక్రియలకు నాలుగు గుర్తులున్నాయి. కూడికకు '+', తీసివేతకు '-' అనే గుర్తులు మనకు తెలుసు. హెచ్చవేతకు సాధారణంగా 'x' అనే గుర్తును గణితంలో ఉపయోగిస్తాం. కాని ఆ గుర్తు ఇంగ్లీషు అక్షరం x ను పోలి ఉండటంతో కంప్యూటర్ పరిభాషలో హెచ్చవేతకు * అనే గుర్తును ఉపయోగిస్తారు. భాగాహారానికి '/ (forward slash) గుర్తును ఉపయోగిస్తాం. ఈ ప్రక్రియలను 'సి'లో ఏవిధంగా చెయ్యాలో ఇప్పుడు చూద్దాం.

చరరాశులకు ఒక విలువను ఇవ్వాలంటే వాటిని ముందు డిక్లేర్ చెయ్యాలని ఇంతకుముందు పాఠంలో చదివాం. అంటే x అనే చరరాశికి 5 అనే విలువను ఇవ్వాలంటే

```
int x;
```

అని డిక్లేర్ చేసి

```
x = 5;
```

అని విలువను ఇస్తాం.

కాని దీనికి రెండు స్టేట్‌మెంట్‌లు అవసరం లేకుండా నేరుగా

```
int x = 5;
```

అని కూడా డిక్లేర్ చేయవచ్చు.

ఇంతకు ముందు పాఠంలో కూడిక గురించి నేర్చుకున్నాం. ఇప్పుడు తీసివేతను చూద్దాం.

```
int x = 5, y = 3, z;
z = x - y;
printf("%d",z);
```

అన్నప్పుడు z లోకి x, y ల తేడా అంటే 5-3 = 2 అనే విలువ వచ్చి చేరుతుంది.

ఇక డిక్లేర్ చేసే సమయంలో కూడా ఈ ప్రక్రియలు చేయవచ్చు.

ఉదాహరణకు

```
int x = 10;
int y = x + z;
```

అని కాని,

```
int y = x – 5;
```

అనికాని డిక్లేర్ చేయవచ్చు. అయితే దీనికి ఒక షరతు ఉంది. x అనే చరరాశిని y కంటే ముందే డిక్లేర్ చేయాలి
ఇప్పుడు హెచ్చవేతకు సంబంధించిన ప్రక్రియను చూద్దాం.

```
int x;

x = 2 * 5;
```

అనే రాస్తే x లో 10 అనే విలువ వచ్చి చేరుతుంది.

ఈ కింది ప్రోగ్రాంను పరిశీలించండి.

```
#include<stdio.h>
main()
{
    int x,y=3, z = 2;
    x = y*z;
    printf("The product of %d and %d is %d", y, z, x);
}
```

ఇప్పుడు దీన్ని రన్ చేస్తే

The product of 3 and 2 is 6

అని ప్రింటవుతుంది.

దీన్నే మనం ఇంకో విధంగా రాయవచ్చు.

```
#include<stdio.h>
main()
{
    int y = 3, z = 2;
    printf("The product of %d and %d is %d", y, z, y*z);
}
```

ఇప్పుడు కూడా

The product of 3 and 2 is 6 అనే ప్రింటవుతుంది.

అంటే గణిత ప్రక్రియలను printf స్టేట్‌మెంట్‌లో రాసిన ప్రక్రియ పూర్తయి ఫలితం ప్రింటవుతుందన్నమాట.

ఇప్పుడు మరో ఉదాహరణను పరిశీలిద్దాం.

ఒక చతురస్రానికి వైశాల్యం కనుక్కునేందుకు ప్రోగ్రాం రాయాలి.

```
#include<stdio.h>
main ()
{
    int side, area;
    side = 5;
    /* Now calculate the area */
    area = side * side;
    printf ("The area of the sqaure is %d \n", area);
    /* End of program */
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి రన్ చేస్తే

The area of the square is 25

అని ప్రింటవుతుంది.

ఒక ఉద్యోగికి సంబంధించిన వివిధ వేతనాలను తీసుకుని నికర వేతనం కనుగొనే ప్రోగ్రాం చేద్దాం.

```
#include<stdio.h>
main ()
{
    int basic, ta, da, hra, net;
    /*Now assign values to variables*/
    basic = 3000;
    ta = 1000;
    da = 1500;
```

```
        hra = 1500;
        net = basic + ta+ da + hra;
        /*print net salary*/
        printf("Net salary of employee is Rs. %d\n", net);
        /*End of program*/
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే తెరపై

Net salary of employee is Rs. 7000

అని ప్రింటవుతుంది.

ఒక సంఖ్యకు దశాంశ స్థానాలు కూడా ఉండవచ్చు. సంఖ్యలో దశాంశ స్థానానికి ముందు అంటే ఎడమవైపు ఉన్న దాని విలువను పూర్ణ భాగం (Integral part) అని, తర్వాత ఉన్న విలువలను వాస్తవ భాగం (real value) అని అంటారు. ఉదాహరణకి

10.642 అన్న సంఖ్యలు

10 Integer part అవుతుంది.

.462 ను real part అంటారు.

మనం చరరాశిని int చరరాశిగా డిక్లేర్ చేస్తే అది కేవలం integer part ను మాత్రమే నిలవ చేసుకుంటుంది. అంటే,

```
int x;
x = 10.462;
```

అని రాస్తే, x లోకి కేవలం 10 మాత్రమే వస్తుంది.

కింది ప్రోగ్రాంను పరిశీలించండి

```
#include<stdio.h>
main ()
{
        /*declare an integer */
        int number;
        /*assign a real number to the variable*/
        number = 12.716;
        /*Now let in print the value in the memory*/
```

```
        printf("The value in the memory of integer is %d", number);
        /* End of program*/
    }
```

ఈ ప్రోగ్రాంను రన్ చేస్తే తెరపై

The value in the memory of integer is 12

అని వస్తుంది.

దీన్ని బట్టి int గా డిక్లేర్ చేసిన చరరాశులు దశాంశ స్థానం ముందున్న సంఖ్యనే నిలవ ఉంచుకుంటాయని తెలుస్తోంది.

దశాంశ స్థాయి తర్వాత ఉన్న విలువలను నిలవ ఉంచుకోడానికి float రకం చరరాశులను ఉపయోగిస్తాం.

```
float x;
x = 12.716;
```

అని డిక్లేర్ చేస్తే x లో పూర్తి విలువ నిలవ ఉంటుంది.

కింది ఉదాహరణలో పరిశీలించండి.

```
#include<stdio.h>
main ()
{
        float real_num;
        real_num = 248.1746;
        /*Now let us print the value*/
        printf("The value in float variables is %f", real_num);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

The value in float variable is 248.1746

అని ప్రింటవుతుంది.

ఇప్పుడు ఒక వృత్త వ్యాసార్థం తీసుకుని వృత్త వైశాల్యం కనుగొనే ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main ()
{
```

```
        float radius, area;
        radius = 5;
        /*area of circle is pi*radius*radius*/
        /*value of pi = 3.14*/
        area = 3.14*radius*radius;
        printf("the area of circle is %f\n", area);
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే

The area of circle is 78.5

అని ప్రింటవుతుంది.

ఇక భాగహారం చూద్దాం. ఈ ప్రోగ్రాం పరిశీలించండి.

```
#include<stdio.h>
main()
{
        int rem, num_1 = 20, num_2 = 10;
        rem = num_1/num_2;
        printf("The result is %d", rem);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

The result is 2

అని ప్రింటవుతుంది.

ఈ భాగహారం చేసేటప్పుడు కొన్ని అంశాలను దృష్టిలో పెట్టుకోవాలి.

i. మనం rem ను int గా డిక్లేర్ చేశాం కాబట్టి దశాంశ స్థానాలు రావు

అంటే num_1=1, num_2 = 2 అని తీసుకుని

rem = num_1/num 2 అని రాశామనుకోండి.

మామూలుగా అయితే ఫలితం

rem = 1/2 = 0.5 అని రావాలి. కాని

rem పూర్ణాంకం (integer) అయినందున

rem = 0 అవుతుంది. ఆ తర్వాత విలువలు Truncate అవుతాయి.

బారువడ్డీ (simple interest) కనుగొనేందుకు ఒక ప్రోగ్రాం రాద్దాం.

```
#include<stdio.h>
void main ()
{
        float p, n, r, i;
        /*p for principle
        n for number of years
        r rate of interest
        i is for interest*/
        p = 1000;
        n = 5;
        r = 6;
        i = (p*n*r)/100;
        /*the formula for simple interest is pnr/100 */
        /*now let us print the value of i */
        printf("the interest for Rs. 1000 for 5 years at the rate of ");
        printf("6 percent per annum is %f\n"), i);
}
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

The interest for Rs. 1000 for 5 years at the rate of 6 percent per annum is 300.

అని ప్రింటవుతుంది.

ఇప్పటివరకూ ఒక చరరాశి విలువను ఇంకో చరరాశికి ఇవ్వాలంటే = గుర్తును ఉపయోగించి assign చేస్తున్నాం. ఉదాహరణకు

```
int = x, y;
x = 3;
y = x;
```

అని రాస్తే x లోని విలువ y లోకి కూడా వస్తుంది.

పై ఉదాహరణల్లోనూ, ఇప్పటివరకూ చేసిన ప్రోగ్రాముల్లోను మనం ఒకే డేటాటైప్‌కు చెందిన చరరాశులనే ఒకదానికొకటి assign చేస్తున్నాం. అదే వేర్వేరు డేటాటైప్ చరరాశులను ఒకదానికొకటి assign చేయవచ్చా? అంటే వేర్వేరు డేటాటైప్‌లు ఉన్న చరరాశుల మధ్య విలువలను మార్చవచ్చా? చేయవచ్చు. అయితే ఇందుకు కొన్ని నిబంధనలున్నాయి.

ముందుగా పెద్ద చరరాశి (అంటే ఎక్కువ మెమొరీని తీసుకునేది)కి చిన్న చరరాశి (తక్కువ మెమొరీని తీసుకునేది) విలువను తేలికగా assign చేయవచ్చు. అయితే ఏ డేటాటైప్ చరరాశి విలువ మెమొరీని ఆక్రమిస్తుంది అనే విషయం ముందుగా తెలియాలి. అందుకు ఈ కింది టేబుల్‌ను పరిశీలించండి.

డేటాటైప్	మెమొరీ
char	1 byte
int	2 bytes
long int	4 bytes
float	4 bytes
double	8 bytes
long double	10 bytes

ఇప్పుడు float డేటాటైప్‌లో ఉన్న చరరాశికి డేటాటైప్‌లో ఉన్న చరరాశిలోని విలువను తేలిగ్గా ఇవ్వవచ్చు.

```
int x = 3;
float y;
y = x;
```

అని ఇచ్చినా సరిపోతుంది.

అయితే ఈ కింది సందర్భంలో తేడా వస్తుంది.

```
int x = 3, y = 8;
float p;
p = y/x;
```

ఈ విధంగా రాసినప్పుడు కుడిచేతిపక్క మనం ఉపయోగించిన రెండు చరరాశులూ int డేటాటైప్‌కు సంబంధించినవే కాబట్టి వాటి భాగాహారం ఫలితం కూడా int డేటాటైప్‌కు అనుగుణంగానే ఉంటుంది. అంటే, మామూలుగా 8/3 = 2.933 అని రావాలి కాని రెండు చరరాశులు int కు సంబంధించినవే కావటంతో భాగాహార ఫలితం 2 అవుతుంది. ఎడమ చేతిపక్క float ను ఉంచినా అందులోకి కేవలం 2 మాత్రమే చేరుతుంది. అదే ఈ రెండు int చరరాశుల భాగహారం జరిగే స్టేట్‌మెంట్ డేటాటైప్‌ను మారిస్తే మనకు p లో 2.933 ప్రత్యక్షమవుతుంది. భాగహారం జరిగే సమయంలో డేటాటైప్‌ను ఈ కింది విధంగా మారుస్తారు.

```
p = (float) y/x;
```

ఏ సమీకరణానికి డేటాటైప్ మార్చలనుకున్నామో దానిముందు ఆ డేటాటైప్‌ను బ్రాకెట్‌లో రాస్తే సరిపోతుంది. దాన్ని ఈ కింది విధంగా చూపవచ్చు.

```
variable = (datatype) expression;
```

ఈ విధంగా assign అయ్యే విలువ డేటాటైప్‌ను మార్చటాన్ని టైప్ క్యాస్టింగ్ (Type casting) అంటారు. ఈ కింది ప్రోగ్రాంను పరిశీలించండి. ఇందులో టైప్ క్యాస్టింగ్ చేసే ముందు ఫలితం, చేసిన తర్వాత ఫలితం ఏవిధంగా ఉంటాయో తెలస్తుంది.

```
#include<stdio.h>
main ()
{
        int x = 5, y = 12;
        float r;
        printf ("The result before type casting\n");
        r = y/x;
        printf ("is %f \n", r);
        print f("The result after type casting \n");
        r = (float)y/x
        print f("is %f \n", r);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

The result before type casting

is 2

The result after type casting is 2.4

అని ప్రింటవుతాయి. దీన్ని బట్టి type casting ప్రాముఖ్యం మనకి తెలుస్తుంది.

భాగహారంలో రెండో సంఖ్య హారం. అంటే పై ఉదాహరణలో num_2 ఎప్పుడూ 0(సున్నా) కాకుండా జాగ్రత్త పడాలి. అది సున్నా అయితే భాగహారం ఈ విధంగా ఉంటుంది. rem = 1/0 ఇది అనిర్వచిత (underfined) విలువ. ప్రోగ్రాం మనకు ఖచ్చితంగా ఎర్రర్ ఇస్తుంది. దీన్ని zero division error అంటారు.

ఇక నాలుగు గణాంక ప్రక్రియలను ఉపయోగించి రాసిన ఈ కింది ప్రోగ్రాంలను పరిశీలించండి.

```
# include<stdio.h>
main()
{
    int num_1, num_2, result;
```

```
num_1 = 20;
num_2 = 10;
result = num_1 + num_2;
printf("sum of %d and %d is %d", num_1, num_2, result)
result = num_1 –num_2;
printf("Difference of %d and %d is %d", num_1, num_2, result);
result = num_1 * num_2;
printf ("product of %d and %d is %d\n"num_1, num_2, result);
result = num_1/num_2;
printf("The result of division of %d by %d is %d", num_1, num_2, result);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే కింది విధంగా ప్రింటవుతుంది.

Sum of 20 and 10 is 30

Difference of 20 and 10 is 10

Product of 20 and 10 is 200.

The result of division of 20 by 10 is 2

గణిత ప్రక్రియల ప్రాధాన్యక్రమం

ఒక ప్రోగ్రాంలో

x = 2 + 3 * 5 – 10 + 2/2 – 1

అని ఉందనుకోండి. దాని ఫలితం ఏవిధంగా ఉంటుంది ?

దీని ఫలితాన్ని తెలుసుకోవాలంటే అసలు నాలుగు ప్రక్రియల్లో ఏ ప్రక్రియ ముందు జరుగుతుంది ? ఈ ప్రక్రియలు ఏ క్రమంలో జరుగుతాయి అన్న విషయం తెలియాలి. 'సి' భాషలో ముందుగా హెచ్చవేత, భాగహారం ఆ తర్వాత కూడిక, తీసివేతలు జరుగుతాయి. ఇప్పుడు పైన పేర్కొన్న లెక్కకు ఫలితం ఈ విధంగా ఉంటుంది.

ముందు 3 * 5 = 15 జరుగుతుంది. అప్పుడు

2 + 3 * 5 – 10 + 2/2 – 1

x = 2 + 15 – 10 + 2/2 – 1 తర్వాత

2 + 15 – 10 + 2/2 – 1

2/2 = 1 అవుతుంది. దాంతో

$x = 2 + 15 - 20 + 1 - 1$

ఇప్పుడు x = 3 అని తేలిగ్గా చెప్పవచ్చు.

ఇందులో మనం గమనించాల్సిన విషయం మరొకటుంది.

$x = (2 + 3) * 5 - 10 + 2/2 - 1$ అని ఇచ్చామనుకోండి. పైన ఉన్న క్రమం మారుతుంది. ముందుగా బ్రాకెట్‌లో ఉన్న ప్రక్రియ జరుగుతుంది. ఆ తర్వాతే బయట ప్రక్రియలు ప్రారంభమవుతాయి. అంటే

$x = 5 * 5 - 10 + 2/2 - 1$గా మారుతుంది. అప్పుడు విలువ x = 15 అవుతుంది. ఈ తేడాను గమనించటం ప్రధానం.

ఈ గుర్తుల ప్రాధాన్యక్రమాన్ని వివరించే పట్టికను కింద పరిశీలించండి. పట్టికలో పైనుంచి కిందకు ప్రాధాన్యక్రమాన్ని అవరోహణ క్రమంలో చూడాలి.

Operator Type	Operators
Unary operators	–, ++, – –, !
Multiply, divide, remainder	*, /, %
add and subtract	+, –
relational operators	<, <=, >, >=
equality operation	==, !=
logical AND	& &
logical OR	//
conditional operator	?:

పై పట్టికలో కొన్ని గుర్తులు మనకు ఇంక తారసపడలేదు. ముందు పాఠాల్లో వాటిని చూద్దాం.

భాగహారం చేసేటప్పుడు శేషం వస్తుంది. ఉదాహరణకు

x = 5/3 అంటే x = 1 అని తెలుసు. దానికి 2 శేషం వస్తుంది. ఈ శేషాన్ని కనుగొనాలంటే 'సి' మనకు ప్రత్యేకంగా ఒక ప్రక్రియను, గుర్తును సమకూరుస్తోంది. 'సి' భాషలో % గుర్తును శేషాన్ని కనుగొనడానికి వాడతారు.

x = 5%3 అని రాస్తే x లోకి ఆ రెండింటి భాగహార ఫలితంగా వచ్చే శేషం చేకూరుతుంది. x విలువ 2 అవుతుంది.

ఈ కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int x, y, z;
    x = 10,
    y = 7;
    z = x%y;
```

```
        printf("The remainder is %d", z);
    }
```

దీన్ని రన్ చేస్తే

The remainder is 3

అని ప్రింటవుతుంది.

ఈ పాఠంలో నేర్చుకున్న వాటిని ఉపయోగించి కొన్ని ప్రోగ్రాంలు ఇప్పుడు చేద్దాం.

(1) కొన్ని అంకెలను తీసుకుని వాటి సగటును కనుక్కుందాం. సగటును కనుక్కోవాలంటే ముందు ఆ అంకెలన్నిటినీ కూడాలి. ఎన్ని అంకెలున్నాయో చూసి ఆ సంఖ్యచేత ఈ మొత్తాన్ని భాగించాలి. $x = 2, y = 4, z = 6$ అయితే వీటి సగటు $\frac{2+4+6}{3}$ అవుతుంది. అంటే 4 అవుతుంది. దీని ప్రోగ్రాం ఈ కింది విధంగా ఉంటుంది.

```
#include<stdio.h>
main()
{
    int x, y, z, average, sum;
    x = 2;
    y = 4;
    z = 6;
    sum = (x+y+z);
    average = sum/3;
    printf("Average of %d, %d and %d is %d", x, y, z, average);
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే ఈ కిందివిధంగా ప్రింటవుతుంది.

average of 2, 4 and 6 is 4.

ఇప్పుడు % గుర్తును ఉపయోగించి ఒక తమాషా ప్రోగ్రాంను చూద్దాం.

```
#include<stdio.h.>
main()
{
    int x, y, z;
    x = 54;
```

```
    y = x%10;
    z = x/10;
    printf("The reverse of %d is %d%d", x, y, z);
}
```

దీన్ని రన్‌చేస్తే

The reverse of 54 is 45

ఇక్కడ 54 మాత్రమేకాక ఏ రెండంకెల సంఖ్య ఇచ్చిన అది వెనక్కి మారుతుంది. ఇది ఎలా సాధ్యపడింది..... మీరే ఆలోచించి కనుక్కోండి.

అభ్యాసం

1. ఉష్ణోగ్రతను ఫారిన్‌హీట్‌లోను, సెంటిగ్రేడ్‌లోను కొలుస్తాము. ఈ రెంటికీ మధ్య ఉన్న సంబంధాన్ని ఈ కింది సమీకరణం ద్వారా చూపవచ్చు.

   C = (F–32)*5/9 - ఫారిన్ హీట్ ఇచ్చి సెంటీగ్రేడ్ కనుగొనటానికి ఇది ఉపయోగిస్తుంది.

   F = 9/5*C+32 - సెంటిగ్రేడ్ ఇస్తే దీని ద్వారా ఫారిన్ హీట్ కనుక్కోవచ్చు.

   ఇప్పుడు C కి 38 40 39 50 వంటి విలువలను తీసుకుని F ను ప్రింట్ చేయడానికి ప్రోగ్రాములు రాయండి. అదేవిధంగా F కి 90, 98, 104, 110, 100 విలువలకి C ఏమవుతుందో ప్రింట్ చేయడానికి ప్రోగ్రాములు రాయండి.

2. 400 రోజులు తీసుకుని అందులో ఎన్ని సంవత్సరాలు, ఎన్ని నెలలు, ఎన్ని వారాలు, ఎన్ని రోజులున్నాయో కనుక్కునేందుకు ప్రోగ్రాం రాయండి.

   (ఈ ప్రోగ్రాం వెంటనే రాకపోయినా కొంత ప్రయత్నించండి: చిన్న క్లూ : గుర్తును % ఉపయోగించాలి)

అనుబంధం

సిలో దశాంశ స్థాయి (Decimals)

ఇప్పటివరకూ మనం Integer అనే డేటాటైప్‌ను పరిశీలించాం. అయితే ఇందులో 0, 1, 2...... వంటి సంఖ్యలలో గణిత ప్రక్రియలు చేయడాన్ని మాత్రమే చూశాం. అయితే దశాంశ స్థానాలు, ఘనాలు లేకపోతే గణితమే లేదు. ఈ విషయం సుస్పష్టం.

ఇప్పుడు దశాంశ స్థానాలు, ఘనాలు (powers) ఉన్న డేటాటైప్‌ను పరిశీలిద్దాం.

ఇంతకుముందే ఈ డేటాటైప్ మనకు పరిచయమైంది. ఇప్పుడు దీన్ని మరింత వివరంగా చూద్దాం. ఈ డేటాటైప్‌లో చరరాశుల డిక్లరేషన్‌కు float అన్న పదాన్ని వాడుతారు.

ఉదాహరణకు

float number;

ఇప్పుడు number అనే చరరాశిలో 4.32, 10.42, 334.6752 వంటి విలువలను నిలవ చేయవచ్చు. సాధారణంగా float లో నిలువచేసే విలువలకు దశాంశ స్థానం తర్వాత ఆరు అంకెలవరకు ఉండవచ్చు. అంటే 10.47525 వంటి విలువలను కూడా నిలువచేయవచ్చు. int ద్వారా నిలువచేసే చరరాశికి మనం (–) లోను, (+) లోను విలువ ఇవ్వవచ్చని తెలుసుకున్నాం కదా. ఇక int ద్వారా డిక్లేర్ చేసిన చరరాశిలో మనం -32, 768 నుంచి 32767 వరకు విలువలను నిలువ చేయవచ్చు. అంటే int ద్వారా డిక్లేర్ చేసిన చరరాశికి ఇచ్చే విలువ -32768, 32767 కి మధ్య ఉండాలి.

ఇంతకన్నా ఎక్కువ విలువ ఉన్న సంఖ్యలను నిలువ చేయడానికి 'సి'లో వీలులేదా? అన్ని అనుమానం మీకు వచ్చింది కదూ!

ఇంతకన్నా ఎక్కువ విలువలను నిలువ ఉంచడానికి 'సి'లో long అనే డేటాటైప్ ఉంది. int లో నిలువ సామర్థ్యాన్ని పెంచడానికి long అనే qualifier ను ఉపయోగిస్తారు. అంటే 64,767 అన్న విలువను x అనే చరరాశిలో నిలువ చేయడానికి

long int x అని డిక్లేర్ చేయాలి.

float ద్వారా మనం దశాంశ స్థానం తర్వాత ఆరు అంకెల వరకు పొందవచ్చని తెలుసుకున్నాం కదా! అదే అంతకన్నా ఎక్కువ కావాలంటే మనం double అనే డేటాటైప్‌ను ఉపయోగిస్తాం. ఇందులో దశాంశ స్థానం తర్వాత 16 అంకెలను మనం ఉండవచ్చు. విజ్ఞాన, అంతరిక్ష శాస్త్రాలకు సంబంధించిన ప్రోగ్రాములు రాసేటప్పుడు ఈ విధమైన ఖచ్చితమైన సమాచారం అవసరమవుతుంచి. అప్పుడు double ను ఉపయోగిస్తాం.

ఇప్పటి వరకూ మనం అంకెల గురించి చర్చించాం. అయితే కంప్యూటర్‌లో అంకెలతో పాటు అక్షరాలు కూడా ఉండాలి. వాటిని ప్రోగ్రాముల్లో ఏవిధంగా ఉపయోగించాలో ఇప్పుడు చూద్దాం.

అక్షరాలను, గుర్తులను నిలువ చేయడానికి మనకు char అనే డేటాటైప్ ఉంది. char అనే పదం character అని సూచిస్తుంది. character అంటే ఒక అక్షరం లేదా అంకె లేదా ఒక గుర్తు కావచ్చు. దీనికి డిక్లరేషన్ ఈ విధంగా ఉంటుంది.

```
char name;
name = 'A';
```

అంటే name అనే చరరాశిలో A అనే అక్షరం ఉంది.

పైన మనం చూసిన డేటాటైప్‌ను ప్రోగ్రాంలలో ఎలా ఉపయోగించాలో ఇప్పుడు చూద్దాం.

```
#include<stdio.h>
main()
{
    float, a, b, c, sum, avg;
    a = 33;
    b = 25;
```

```
        c = 15;
        sum = a+b+c;
        avg = sum/3;
        printf("The average of %f, %f and %f is %f"; a, b, c, avg.);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

The average of 33, 25 and 15 is 25.333333 అని ప్రింటవుతుంది.

ఇక్కడ గమనించారా! printf స్టేట్‌మెంట్‌లో %d కి బదులుగా %f వచ్చింది. అంటే float చరరాశులలోని విలువలను ప్రింట్ చేయాలంటే %f ఉపయోగిస్తాము. అదే long int ని ఉపయోగిస్తే వాటిని ప్రింట్ చేయడానికి %ld ని ఉపయోగిస్తాం. ఇక char ద్వారా డిక్లేర్ చేసిన చరరాశులను %c ఉపయోగించటం ద్వారా ప్రింట్ చేయవచ్చు.

```
#include<stdio.h>
main()
{
        char x1, x2, x3, x4;
        x1 = 'V';
        x2 ='E';
        x3 = 'N';
        x4 = 'U';
        printf ("%c%c%c%c", x1, x2, x3, x4);
}
```

దీన్ని రన్ చేస్తే VENU అని ప్రింటవుతుంది. వరసగా నాలుగు అక్షరాలను ఒకదాని తర్వాత మరొకటి ప్రింట్ చేయడంతో ఆ అక్షరాల వరస ఒక పదంలాగా కనిపిస్తుంది. ఒక్కడ ఒక విషయం గుర్తించండి. అక్షరాలను చరరాశులకి ఇచ్చేటప్పుడు (' ') సింగిల్ కొటేషన్ మార్కుల మధ్య ఉంచాలి.

char name;

name = a అనటం తప్పు

name = 'a' అని రాయాలి.

అక్షరాలనే కాక #, %, : వంటి గుర్తులను కూడా మనం ఈ char చరరాశులలో నిలువ చేయవచ్చు. అదే విధంగా మనం ఇంతకుముందు చూసిన '\n' అనే గుర్తును కూడా నిలువ చేయవచ్చు.

ఇప్పుడు ఈ తమాషా ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
        char C1, C2, C3, C4, C5;
```

```
    C1 = 'R';
    C2 = 'A';
    C3 = 'M';
    C4 = 'U';
    C5 = '\n';
    printf("%C%C%C%C%C%C%C", C1, C5,C2,C5,C3,C5,C4);
}
```

దీన్ని రన్ చేస్తే ఈ విధంగా ప్రింటవుతుంది.

R

A

M

U

ఎందుకంటే మనం printf లో ప్రతి అక్షరం తర్వాత '\n' ని ప్రింట్ చేస్తున్నాం. మనం రాసేది

R\nA\nM\nU

అని రాస్తున్నాం.

ASCII కోడ్ : కీబోర్డు ద్వారా అక్షరాలను, గుర్తులను కంప్యూటర్‌లోకి ఇన్‌పుట్‌గా ఇచ్చి వాటిని మెమొరీలో భద్రపరుస్తున్నాం. మెమొరీలో ఏ గుర్తు అయినా, అక్షరం అయినా, అంకె అయినా ఒక సంఖ్య రూపంలోనే నిలవ ఉంటుంది. కీబోర్డుపై ఉన్న అన్ని అక్షరాలు, గుర్తులతో పాటు కొన్ని ప్రత్యేక గుర్తులకు కూడా ఒక కోడ్ సంఖ్యను కేటాయించారు. ఉదాహరణకు A అనే అక్షరానికి కోడ్ సంఖ్య 65. అదే a అనే అక్షరానికి కోడ్ సంఖ్య 97. ఇదే విధంగా అన్ని అక్షరాలకు, అంకెలకూ కోడ్ సంఖ్యలు ఉన్నాయి.

కీబోర్డుపైగల అంకెలకు కూడా కోడ్ సంఖ్యలు ఉన్నాయి. ఉదాహరణకు 0 కి కోడ్ విలువ 45 కాగా 9 కి కోడ్ విలువ 54. ఈ రకంగా కోడ్ వాడటం వల్ల అన్ని రకాల గుర్తులను మనం సంఖ్యల రూపంలో నిలవచేసుకోవచ్చు. ఏ గుర్తులకు ఏ కోడ్ వాడాలనే అంశం చాలా ప్రధానమైనది. ఒక కంప్యూటర్‌లో A కి 65 అనే కోడ్‌ను ఇచ్చి మరో కంప్యూటర్‌లో 85 అనే విలువను ఇస్తే అయోమయ పరిస్థితి ఏర్పడుతుంది. దీన్ని అధిగమించడానికి ఒక అంతర్జాతీయ కోడ్‌ను రూపొందించారు. దీన్నే ASCII (American Standard Code for Information Interchange) (ఆస్కి) అంటారు. ఈ కోడ్‌ను ఉపయోగించటం వల్ల మన ప్రోగ్రాంను ఏ కంప్యూటర్‌పై ఉపయోగించినా ఒకే విధమైన ఫలితాలు వస్తాయి. ఆస్కి ప్రకారం ఏ గుర్తుకు ఏ కోడ్ సంఖ్య ఉంటుందో ఈ పుస్తకం చివర ఇచ్చాం పరిశీలించండి. అన్ని కోడ్‌లను ఉపయోగించి చేసే కొన్ని ప్రోగ్రాంలను ఈ కింద చూద్దాం.

```
#include<stdio.h>
main ()
{
    /* Examples for ASCII characters */
    int num1, num2;
```

```
        num1 = 65;
        /* ASCII code for A */
        num2 = 97;
        /* ASCII code for a */
        /* Now let us print the characters */
        printf("The characters are %c %c \n", num1, num2);
}
```

ఈ ప్రోగ్రాంలను ఎగ్జిక్యూట్ చేస్తే A, a లు ప్రింటవుతాయి.

కింది ప్రోగ్రాంలో Upper case అక్షరాలను Lower case గా మారుస్తున్నాం. పరిశీలించండి.

```
#include<stdio.h>
main ()
{
        /* Converting upper case letters to lower case */
        int i;
        char let_1, let_2, let_3, let_4, let_5;
        let_1 = 'H';
        let_2 = 'E';
        let_3 = 'L';
        let_4 = 'L';
        let_5 = 'O';
        /* The ASCII code for each lowercase letter is 32 more than its uppercase. so we
have to add 32 to each variable */
        let_1 = let_1 + 32;
        let_2 = let_2 + 32;
        let_3 = let_3 + 32;
        let_4 = let_4 +32;
        let_5 = let_5 + 32;
        /* Now let us print the characters */
        printf("%c%c%c%c", let_1, let_2, let_3, let_4, let_5);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

hello

అని output వస్తుంది.

✦✦✦

అధ్యాయం 6

నియమాలు (Conditions)

"వాన వస్తే టివీలో క్రికెట్ చూద్దాం"

"వాన రాకపోతే గ్రౌండ్‌లో క్రికెట్ ఆడదాం"

అని మీ ఫ్రెండ్ మీకు చెప్పారనుకోండి. మీ నిర్ణయం దేనిపై ఆధారపడి ఉంటుంది? వాన వచ్చేది, రానిది అనే విషయంపై కదా! క్రికెట్ చూడాలా, ఆడాలా అన్న నిర్ణయం ఆ సమయం వచ్చేదాకా తీసుకోరు. ఆ సమయానికి వాన వస్తుందో రాదో పరిశీలించి నిర్ణయం తీసుకుంటాం. అదేవిధంగా కంప్యూటర్ కూడా కొన్ని అంశాల్లో నిర్ణయాలు తీసుకుంటుంది. కొన్ని లెక్కలు చేస్తే వచ్చే ఫలితాన్ని బట్టి నిర్ణయం మారుతుంటుంది.

ఉదాహరణకు ఫలితం 7 కంటే ఎక్కువయితే ఫలితాన్ని రెట్టింపు చేయాలి,

ఫలితం 7 కంటే తక్కువయితే ఫలితాన్ని సగం చేయాలి,

ఫలితం 7తో సమానమయితే OK అని ప్రింట్ చేయాలి.

ఈ మూడింటిలో దేన్ని ఎంచుకోవాలి అన్న నిర్ణయం లెక్క ఫలితంపై ఆధారపడి ఉంటుంది. ఆ ఫలితం తెలిశాక ముందు మొదటి లైను తర్వాత రెండు ఆపై మూడో లైన్లను పరిశీలించి అందులో ఏది సరిపోతుందో నిర్ణయించాలి.

కంప్యూటర్ ఎక్కువ తక్కువలను నిర్ణయించాలంటే అందుకు తగిన సాధనలుండాలి. వీటినే రిలేషనల్ ఆపరేటర్స్ (relational operators) అంటారు. రెండు సంఖ్యలను పోల్చడానికి వీటిని ఉపయోగిస్తారు. ఇవి సత్యము (TRUE), అసత్యము (FALSE) అనే రెండు విలువలనిస్తాయి.

$7 > 4 \rightarrow$ TRUE

$8 < 3 \rightarrow$ FALSE

$9 = 3 \rightarrow$ FALSE

$6 = 6 \rightarrow$ TRUE వంటి విలువలు వస్తాయి.

రెండు చరరాశులనూ పరిశీలించవచ్చు.

$x = 2$

$y = 4$ అనుకుంటే

$x = 2 \rightarrow$ TRUE

x > 4 →FALSE అవుతాయి కదా.

ఇక x, y ల సంగతి మనకు తెలియదనుకోండి. అప్పుడు 'ఇలా జరిగితే' (if) అన్న నిబంధనను వాడతాం. మరో మాటలో చెప్పాలంటే y కన్నా x పెద్దదయితే x కి రెండు కలుపు. దీన్నే మరో విధంగా

if x > y అయితే x కి రెండు కలుపు అని రాయవచ్చు.

దీన్నే 'సి' పరిభాషలో

```
if (x>y)
x = x+2;
```

అని రాస్తాం. ఇక్కడ if అనే పదం పక్కన మనకు కావలసిన నియమాన్ని బ్రాకెట్లలో ఇస్తాం. క్రిందిలైన్‌లో (పక్కన కూడా ఇవ్వవచ్చు) అలా జరిగితే ఏం చెయ్యాలో రాశాము. x = x+2 అంటే x లో ఇప్పటికే ఉన్న విలువకు రెండు కలపటమని అర్థం. ఇక్కడ ఒక విషయం గమనించండి. if పక్కన షరతు లేదా నిబంధన ఉంది. దాని పక్కన సెమికోలన్ లేదు. అదే కింద లైన్‌లో సెమికోలన్ ఉన్నది.

ఇక y కంటే x పెద్దదయితే అనడానికి గణితంలోని గుర్తు >(Greater Than) ను వాడాం. అవే గుర్తులను యధాతధంగా వాడవచ్చు. y కంటే x చిన్నదైతే అనడానికి if (x<y) అని రాయవచ్చు.

y కంటే x పెద్దది లేదా సమానమైతే అని రాయడానికి

if (x>=y) అని రాస్తాం.

అదేవిధంగా y కంటే x చిన్నది లేదా సమానమైతే అన్నప్పుడు

if (x<=y) అని ఉపయోగిస్తాం.

ఇక మరి రెండు షరతులున్నాయి.

మొదటిది సమానమయితే అని రెండు సమానం కాకపోతే అని.

సమానమైతే అనడానికి

if (x=y) అని రాయవచ్చా? రాయకూడదు ఎందుకంటే x = y అంటే

y లో విలువ x కి వచ్చి చేరుతోందని అర్థం.

అందువల్ల సమానమైతే అన్న షరతులను పరిశీలించడానికి 'సి' మనకు == అనే గుర్తును ఇచ్చింది. ఇప్పుడు y కి x విలువ సమానమైతే అన్న షరతును

if (x==y) అని రాయవచ్చు. ఇక్కడ మనం == అనే గుర్తులు పక్కపక్కన రాస్తున్నాం. రెండింటి మధ్య స్పేస్ ఉండదు.

ఇక సమానం కాకపోతే అన్న షరతును if (x!=y) అని రాస్తాం. ఇక్కడ ! అనే గుర్తుకు NOT అనే అర్థం ఉంది.

అదే y కి సమానం కాకపోతే - ఎక్కువైన లేదా తక్కువైనా మనకు సంబంధం లేదు. సమానమా కాదా అని చూడటమే వాటి ఉద్దేశం.

```
if (x!=y)
x = x+y
```

అని రాయవచ్చు.

ఈ గుర్తులను పట్టికలో చూద్దాం.

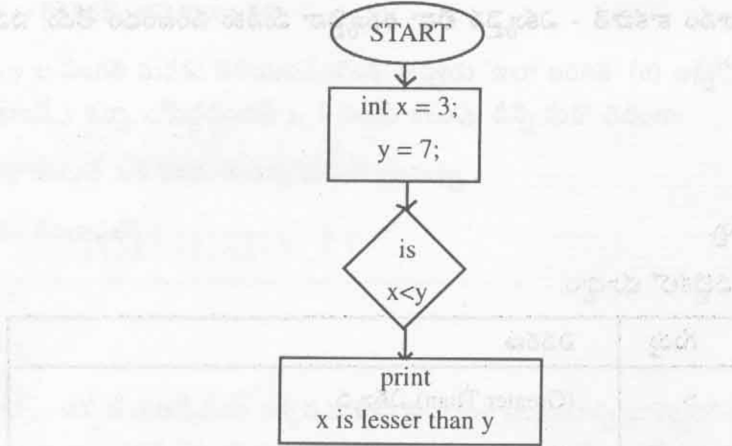
గుర్తు	వివరణ
>	(Greater Than) ఎక్కువ
<	(Less Than) తక్కువ
>=	Greater Than or Equal To ఎక్కువ లేదా సమానమైతే
<=	Less Than or Equal To తక్కువ లేదా సమానమైతే
==	Equal to సమానమైతే
!=	Not Equal To సమానం కాకపోతే

దీనిపై ఒక ప్రోగ్రాంను ఇప్పుడు చూద్దాం.

```
# include<stdio.h>
main()
{
    int x, y;
    x = 3;
    y = 7;
    if (x<y)
    printf("x is lesser than y");
}
```

దీన్ని రన్ చేస్తే x is lesser than y అని ప్రింటవుతుంది.

పైన చూపిన ప్రోగ్రాంను ఈ కింది విధంగా చూపవచ్చు.



పై బొమ్మను Flow chart అంటారు. ఇందులో మన ప్రోగ్రాం ఏ విధంగా ఎగ్జిక్యూట్ అవుతుందో తెలుస్తుంది. ముందుగా x, y చరరాశులకి విలువలు ఇస్తున్నాం. తర్వాత ఏది చిన్నదో పరీక్షించి ఫలితాన్ని ప్రింట్ చేస్తున్నాం. ఈ విధంగా ప్రోగ్రాంలోని ప్రతి చర్యను బొమ్మల రూపంలో చూపే ఫ్లో ఛార్ట్‌లు ప్రోగ్రాం గురించి తెలుసుకోడానికి ఉపయోగపడతాయి.

మరో ప్రోగ్రాంను చూద్దాం. ఇక్కడ రెండు దీర్ఘచతురస్రాల పొడవు వెడల్పులు తీసుకుని వాటిలో దేని వైశాల్యం ఎక్కువో నిర్ణయించే ప్రోగ్రాం పరిశీలించండి.

```
#include<stdio.h>
main()
{
        int len_1, br_1, area_1, len_2, br_2, area_2;
        len_1 = 5;
        br_1 = 3;
        len_2 = 4;
        br_2 = 4;
        area_1 = len_1*br_1;
        area_2 = len_2*br_2;
        /*Now compare areas*/
        if (area_2>area_1)
        printf("The area of bigger rectangle is %d\n", area_2);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

The area of bigger rectangle is 16

ఇంకో అంశం మనం చర్చించాల్సి ఉంది. పైన మనం ఒక షరతును పరిశీలించాం.

ఇలా అయితే - ఏం చెయ్యాలి అన్న షరతును చూశాం.

కాకపోతే - ఏం చెయ్యాలి అని కూడా చెప్పాలి కదా !

అందుకని ELSE అనే పదం వాడతాం. దీనికి 'అలాకాకపోతే' అని అర్థం.

ఉదాహరణకు

```
if (x<5)
x = x+5;
else
x = x–5;
```

పై స్టేట్‌మెంట్‌ల అర్థం గ్రహించే ఉంటారు కదూ!

x అనే చరరాశి విలువ 5 కన్నా తక్కువైతే దానికి 5 కలపాలి. కాకపోతే 5 తీసేయాలి. అంటే x విలువ 5 లేదా అంతకన్నా ఎక్కువయితే అందులోంచి 5 తీసేయాలని అర్థం.

పైన రాసిన దాన్ని ప్రోగ్రాం రూపంలో చూద్దాం.

```
# include<stdio.h>
main()
{
    int x = 3;
    if (x>=5)
    printf("x is greater than five or equal to five");
    else
    print f("x is less than five");
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

x is less than five

అని వస్తుంది.

(సూచన : x విలువను మార్చి ఈ ప్రోగ్రాంను మళ్లీ రన్ చేసి చూడండి)

ఒకటి కాకపోతే మరొకటి అని ఇప్పుడు చూశాం. అది కూడా కాకపోతే అప్పుడేం చెయ్యాలి. దానికి మనం else if అని వాడతాం. ఉదాహరణకి

```
if (x>5)
```

```
printf("x is greater than 5");
else if (x==5)
printf("x is equal to 5");
else
printf("x is less than 5");
```

ఇందులో మనం ముందు x>5 అయితే అన్న షరతును, కాని పక్షంలో x=5 అయితే అన్న షరతును, అదీ కాని పక్షంలో ఇక చివరగా మిగిలిన x<5 ని పరిశీలించాం.

ఒకటి కన్నా ఎక్కువ షరతులను పరిశీలించాల్సి వచ్చినప్పుడు వీటిని వాడతాం. ఒక చోట కేవలం ఒక if మాత్రమే రావచ్చు. కాని else if లు ఎన్నయినా రావచ్చు. ఉదాహరణకు day అనే చరరాశి తీసుకుని అందులో 1 ఉంటే ఆదివారం లేదా 2 ఉంటే సోమవారం .... ఇలా ప్రింట్ చేసే ప్రోగ్రాంను పరిశీలిద్దాం.

```
# include<stdio.h>
main()
{
    int day=3;
    if (day ==1)
    printf("Sunday");
    else if (day==2)
    printf("Monday");
    else if (day==3);
    printf("Tuesday");
    else if (day==4)
    printf("Wednesday");
    else if (day==5)
    printf("Thursday");
    else if (day ==6)
    printf ("Friday");
    else if (day ==7)
    printf("Saturday");
    else
    printf("Not a valid day");
}
```

ఈ ప్రోగ్రాంలో day అనే చరరాశికి 1 నుంచి 7 వరకు ఏ విలువను ఇచ్చినా దానికి సంబంధించిన వారం పేరును ఇస్తుంది. వేరే విలువలు ఏమైనా ఇస్తే Not a valid day అని ఇస్తుంది. day కి వేర్వేరు విలువలు ఇచ్చి ఈ ప్రోగ్రాంను రన్ చేసి చూడండి.

పైన రాసిన దీర్ఘచతురస్రాల వైశాల్యాల ప్రోగ్రాంను మరోసారి else_if ఉపయోగించి చూద్దాం.

```
#include<stdio.h>
main ()
{
        int len_1, br_1, len_2, br_2, area_1, area_2;
        len_1 = 5;
        br_1 = 3;
        len_2 = 4;
        br_2 = 4;
        area_1 = len_1*br_1;
        area_2 = len_2*br_2;
        /*Now compare areas */
        if (area_1>area_2)
{
        printf("Area_1 is greater \n");
        printf ("Its area is %d \n", area_1);
}
else
{
        printf("Area of second is greater \n");
        printf("Its area is %d \n", area_2);
}
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

```
Area of second is greater

Its area is 16
```

అని ప్రింటవుతుంది.

మరో if రకం స్టేట్‌మెంట్ :

ఇప్పటి వరకూ మనం ఒక షరతు లేదా నిబంధనకు సంబంధించిన if స్టేట్‌మెంట్‌ను చూశాం. రెండు మూడు నిబంధనలు ఉంటే దానికి సంబంధించిన if స్టేట్‌మెంట్ ఎలా రాయాలో చూద్దాం.

x > 4 మరియు y > 10 అయితే z = z+10;

అని రాయాలి. దీనికి రెండు నిబంధనలున్నాయి.

x > 4 మరియు y > 10

ఈ రెండు నిబంధనలు 'సత్యము' అయితేనే

z = z+10 అవుతుంది.

ఈ రెండిటిలో ఏ ఒక్క నిబంధన అసత్యమైనా పై స్టేట్‌మెంట్ ఎగ్జిక్యూట్ కాదు. దీనికి మనం if స్టేట్‌మెంట్‌ని

```
if (x > 4  && y > 10)
z = z+10;
```

అని రాస్తాం.

ఇందులో if స్టేట్‌మెంట్ తర్వాత రెండు నిబంధనలనూ రాశాం. అయితే రెండు నిబంధనల మధ్య && (రెండు యాంపర్సెండ్) గుర్తులు ఉన్నాయి. ఈ గుర్తు పైన పేర్కొన్న 'మరియు' (and) ని సూచిస్తుంది.

రెండు నిబంధనలు సత్యమైతేనే ఈ మొత్తం if సత్యమవుతుంది.

దీనికి సంబంధించిన ఈ ఉదాహరణను పరిశీలించండి.

```
# include<stdio.h>
main()
{
    int x = 4,  y = 6,  z;
    if(x>5 && x+y<11)
    z = x*y;
    else
    z = x+y;
    printf("\n The result is %d", z);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

```
The result is 10
```

అని ప్రింటవుతుంది.

పై ప్రోగ్రాంలో x+y<11 అన్న నిబంధన సత్యమే కాని x>5 అన్నది అసత్యం. అందువల్ల z = x+y అన్న స్టేట్‌మెంట్ ఎగ్జిక్యూట్ అవుతుంది.

x, y విలువలను మార్చి ఈ ప్రోగ్రాంను మళ్లీ రన్ చేయండి.

ఇప్పుడు మరో రకమైన if స్టేట్‌మెంట్ పరిశీలిద్దాం.

x>5 లేదా y>7 అయితే z = x+y;

దీన్ని if స్టేట్‌మెంట్‌లో ఈ కింది విధంగా రాయవచ్చు.

```
if (x>5 || y>7)
z = x+y;
```

పై స్టేట్‌మెంట్‌లో s>5 లేదా y>7 రెండు నిబంధనల్లో ఏది సత్యమైనా మొత్తం if స్టేట్‌మెంట్ సత్యమవుతుంది.

పైన పేర్కొన్న if రకంలో రెండు నిబంధనలూ సత్యమైతేనే మొత్తం if సత్యమవుతుంది. రెండోరకంలో కనీసం ఒక నిబంధన సత్యమైనా మొత్తం సత్యమవుతుంది. ఈ కింది ప్రోగ్రాం పరిశీలించండి.

```
# include<stdio.h>
main()
{
    int x, y, z;
    x = 6;
    y = 8;
    if(x<5 || y>7)
    z = x+y;
    else
    z = y–x;
    printf("The value of z is %d", z);
}
```

పై ప్రోగ్రాంను రన్ చేస్తే

z = 14

అని ప్రింటవుతుంది.

ఇక్కడ x < 5 సత్యం కాకపోయినా

y>7

కంప్యూటర్ రంగంలోని అద్భుత ప్రోగ్రామింగ్ పుస్తకం

'సి' నేర్చుకుందాం



కె. కిరణ్ కుమార్

పన్నాల వేణుగోపాల్

సత్యమే కాబట్టి మొత్తం if స్టేట్‌మెంట్ సత్యమవుతుంది.

రెండు వేర్వేరు నిబంధనలు ఒకేచోట ఉన్నప్పుడు ఎలా వ్యవహరించాలో మరో రకం if స్టేట్‌మెంట్‌లో చూద్దాం. ఈ కింది నిబంధనలు చూడండి.

salary>2000 మరియు service>3 అయితే

bonus = 20% of basic

salary>2000 మరియు service>8 అయితే

bonus = 25% of basic

salary>5000 మరియు service>5 అయితే

bonus = 10% of basic

అన్న నిబంధనలున్నాయి. ఈ నిబంధనలను స్టేట్‌మెంట్‌లు ఉపయోగించి

```
if(salary>2000)
{
    if(service>3)
    bonus = basic*20/100;
    else if(service>5)
    bonus = basic*25/100;
}
else (salary>5000 && service>5)
    bonus = basic*10/100;
```

పై స్టేట్‌మెంట్‌లలో if(salary>2000) అన్న నిబంధనను ఒకసారి ఉపయోగించాం. అంటే రెండో నిబంధనను రెండుసార్లు పరీక్షించినా అది మొదటి నిబంధనకు లోబడే ఉంటుంది.

దీన్ని వివరంగా చూస్తే –

ముందుగా salary>2000 నిబంధన సత్యమైతేనే service అనే నిబంధనను చూస్తాం. అదే salary>2000 అసత్యమైతే service కి సంబంధించిన నిబంధనను మనం పరిశీలించము. నేరుగా కింది స్టేట్‌మెంట్‌కు వెళ్తాం. ఈ రకమైన if స్టేట్‌మెంట్ వ్యవస్థని Nested if స్టేట్‌మెంట్‌లు అంటారు. ఇందులో ఒక నిబంధన లోపల మరొకటి కాని, మరికొన్ని నిబంధనలు కాని ఉంటాయి. మొదటిది సత్యమైతేనే మిగిలిన వాటిని పరిశీలిస్తాం. కాని పక్షంలో మనం ఆ మొత్తం నిబంధనలను పరీక్షించాం.

ప్రోగ్రాంకు విలువలను ఇవ్వటం

ఇప్పటి వరకూ మనం చేసిన ప్రోగ్రాముల్లో విలువలను మనం ముందుగానే assign చేస్తున్నాం. అయితే అన్ని

సందర్భాల్లోనూ ముందుగానే విలువలను ఇవ్వటం సాధ్యం కాదు. ఉదాహరణకు దీర్ఘచతురస్రం వైశాల్యాన్ని కనుగొనే ప్రోగ్రాంలో పొడవు వెడల్పులు ముందుగానే ఇస్తున్నాం. అదే పొడవు వెడల్పుల విలువలను మార్చాలంటే ఆ ప్రోగ్రాంను మార్చాలి. నిజానికి ఎటువంటి పొడవు వెడల్పులు ఇచ్చినా వైశాల్యాన్ని కనుగొనే ప్రోగ్రాం మనం రాయాలి. అంటే ప్రోగ్రాం రాసే సమయంలో కాక అది రన్ అవుతున్న సమయంలో విలువలను ఇచ్చే వీలుండాలి.

ప్రోగ్రాం రాయటం పూర్తయిన తర్వాత .C ఎక్స్టెన్షన్‌తో సేవ్ చేస్తాం. కంపైల్ అయిన తర్వాత .EXE అనే ఫైల్ వస్తుంది. ఈ ఫైల్‌ను ఎగ్జిక్యూటబుల్ ఫైల్ అంటాం. ఈ ఫైల్ పేరు కూడా మనం ఇచ్చిన ఫైల్ పేరే ఉంటుంది. ఉదాహరణకు prog.c అన్న పేరుతో ఫైల్‌ను సేవ్ చేస్తే prog.exe అనే ఫైల్ సృష్టించబడుతుంది. డాస్‌లో ఈ ఫైల్ పేరును c:\ దగ్గర ఇవ్వగానే మన ప్రోగ్రాం ఎగ్జిక్యూట్ అవుతుంది. ఈ ఫైల్‌ను మన దగ్గర 'సి' కంపైలర్ లేకపోయినా రన్ చేయవచ్చు. మన కంప్యూటర్‌లో ప్రోగ్రాం రాసి దాన్ని కంపైల్ చేసిన తర్వాత వచ్చే ఎగ్జిక్యూటబుల్ ఫైల్‌ను 'సి' కంపైలర్ లేని వేరే కంప్యూటర్‌లో ఎగ్జిక్యూట్ చేయవచ్చు.

అయితే ఎగ్జిక్యూట్ చేసే సమయంలో చరరాశుల విలువలను మార్చాలంటే కుదరదు కదా! అందుకని ప్రోగ్రాం రన్ చేసే సమయంలో చరరాశులకు విలువలను ఇచ్చే స్టేట్‌మెంట్‌ను మనం ఉపయోగించుకోవాలి. ఈ స్టేట్‌మెంట్‌లనే Input స్టేట్‌మెంట్‌లంటారు. ఈ Input స్టేట్‌మెంట్‌ల ద్వారా చరరాశులకు విలువలను ఇవ్వవచ్చు.

'సి'లో చరరాశికి విలువనివ్వటానికి scanf అనే స్టేట్‌మెంట్‌ను ఎక్కువగా ఉపయోగిస్తారు.

```
scanf("%d", &x);
```

ఇది కూడా దాదాపుగా printf స్టేట్‌మెంట్‌లాగానే ఉంటుంది కానీ ఇక్కడ చరరాశికి ముందు &(యాంపర్సెండ్)ను ఉంచుతాం అదే తేడా.

ఈ కింది ఉదాహరణను చూడండి.

```
# include<stdio.h>
{
    main()
    int x ;
    printf("Please enter value for x \n");
    scanf("%d", &x);
    printf(" You entered %d", x);
}
```

ఈ ప్రోగ్రాంను రన్ చేయండి.

ఇక్కడ ముందుగా

Please enter value for x

అని ప్రింటవుతుంది.

దాని కింద కర్సర్ మీ కోసం ఎదురు చూస్తూ ఉంటుంది. కీబోర్డు ద్వారా ఒక అంకెను లేదా సంఖ్యను టైప్ చేసి Enter కీని నొక్కండి. వెంటనే మీరు ఎంటర్ చేసిన విలువ మళ్లీ వస్తుంది. ఉదాహరణకి మీరు 24 అనే సంఖ్యను ఎంటర్ చేస్తే

You entered 24

అని వస్తుంది. అదే ప్రోగ్రాంను మళ్లీ రన్ చేస్తే మరో విలువను ఎంటర్ చేసే అవకాశం ఉంటుంది.

ఇప్పుడు ఇంకో ప్రోగ్రాం చూద్దాం.

```
# include<stdio.h>
main()
{
    int x, y, z;
    printf("Enter first number");
    scanf("%d", &x);
    printf("Enter second number");
    scanf("%d", &y);
    z = x+y;
    printf("The sum of %d and %d is %d", x, y, z);
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే

Enter first number

5 (అంకెను టైప్ చేసి Enter బటన్ నొక్కాలి)

Enter second number

6

The sum of 5 and 6 is 11

అని ప్రింటవుతుంది.

ఇప్పుడు పైన చూసిన దీర్ఘచతురస్ర వైశాల్యాల ప్రోగ్రాంను మరోసారి చూద్దాం.

```
#include<stdio.h>
main ()
{
        int len_1, br_1, len_2, br_2, area_1, area_2;
        printf("Enter value of length of first rectangle \n");
```

```
scanf("%d", & len_1);
printf("enter breadth of first rectangle \n");
scanf("%d", & br_1);
printf(enter length of second rectangle \n);
scanf("%d", & len_2);
printf("enter breadth of second rectangle \n");
scanf("%d", & br_2);
area1 = len1*br_1;
area2 = len2*br_2;
if (area_1>area_2);
printf("first rectangle is larger \n");
else if ("area_1<area_2)
printf("second rectangle is larger \n");
else
printf ("Both are same in area");
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

Enter the value of length of first rectangle

8(అని టైప్ చేసి బటన్ నొక్కాలి)

Enter breadth of first rectangle

2

Enter length of second rectangle

4

Enter breadth of second rectangle

4

Both are same in area.

ఈ విధంగా మనం లేదా ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసేవారు ఎంటర్ చేసే విలువల ఆధారంగా ఫలితం వస్తుంది.

ఇద్దరు వ్యక్తుల వయసులను తీసుకుని వారిలో ఎవరు పెద్దవారో నిర్ణయించే ప్రోగ్రాం రాద్దాం.

```
#include<stdio.h>
main ()
{
    int age_ram, age_syam;
```

```
        printf("Enter age of Ram\n");
        scanf("%d", f age_ram);
        printf ("Enter age of Syam \n");
        scanf("%d", & age_syam);
        if (age_ram>age_syam)
        printf ("Ram is elder to Syam \n");
        else if (age_ram < age_syam)
        printf("Syam is elder to Ram\n");
        else
        printf ("Both are of same age \n");
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే ఫలితం కింది విధంగా వస్తుంది.

```
Enter age of Ram
25
Enter age of Syam
37
Syam is elder to Ram
```

పైన ప్రోగ్రాంలో ఒక చిన్న ఇబ్బంది ఉంది గమనించండి. else if అనే statements ను మనం ఆరుసార్లు రాయాల్సివచ్చింది. అదే ప్రోగ్రాంను సంవత్సరంలో నెలలను కనుగొనడానికి చేశామనుకోండి. else if స్టేట్‌మెంట్ 12 సార్లు రాయాల్సి వస్తుంది. ఈ ఇబ్బందిని అధిగమించడానికి మనకు 'సి'లో మరో సదుపాయం ఉంది. అదే switch....cas స్టేట్‌మెంట్లు.

స్విచ్ .... కేస్

పై ప్రోగ్రాంను ఈ కింద మరోవిధంగా రాశాం. ఒకసారి గమనించండి.

```
# include<stdio.h>
main()
{
    int week;
    printf("Enter any number to select a week\n");
    scanf("%d",.&week);
    switch(week)
    {
```

```
case 1 : printf("Sunday");
    break;
case 2 : printf("Monday");
    break;
case 3 : printf("Tuesday");
    break;
case 4 : printf("Wednesday");
    break;
case 5 : printf("Thursday");
    break;
case 6 : printf("Friday");
    break;
case 7 : printf("Saturday");
    break;
}
}
```

ఈ ప్రోగ్రాంను రన్ చేసినా పై ప్రోగ్రాంకు వచ్చిన ఫలితమే వస్తుంది. అయితే ఇందులో ప్రోగ్రాం తేలికగా ఉండటంతోపాటు వేగంగా కంపైల్ అవుతుంది.

ప్రధానంగా week అనే చరరాశిపై ఆధారపడే వారాల విలువలు ప్రింటవుతున్నాయి. అందుకనే week ను switch అనే స్టేట్‌మెంట్‌కు ఇచ్చాం. అప్పుడు week లో విలువ కింద ఉన్న case లలో దేనికి సరిపోతుందో అది ఎగ్జిక్యూట్ అవుతుంది. అంటే week కు మనం 3 అనే విలువను ఇస్తే case 3 : లో ఉన్న స్టేట్‌మెంట్‌లు రన్ అవుతాయి. ఈ విధంగా సంఖ్యలు అయితే మనం ఏ వరసలో రావచ్చని ఊహిస్తున్నామో అదే విధంగా case కు ఇవ్వవచ్చు. ఉదాహరణకు switch లో గనుక 10, 20, 30 వంటి విలువలు వచ్చే అవకాశం ఉంటే, case 10 : case 20 : వంటి విలువలను కూడా మనం ఇవ్వవచ్చు.

ఈ స్విచ్.....కేస్‌లో ఒక చిన్న ఇబ్బంది ఉంది. పై ప్రోగ్రాంలో week విలువ 3 అయితే case 1 :, case 2 : అను వదిలేసి case 3 : లో ఉన్న స్టేట్‌మెంట్‌లను ఎగ్జిక్యూట్ చేస్తుంది. అయితే ఆ తర్వాతి case లు అన్నీ వరసగా ఎగ్జిక్యూట్ అవుతాయి. అంటే case 6 :, case 7 : లు కూడా వరసగా రన్ అవుతాయి. దీన్ని తప్పించడానికి మనం break అనే స్టేట్‌మెంట్‌ను వాడాం. ఈ break ను ఉపయోగిస్తే మన షరతులకు సరిపోయే.... case మాత్రమే ఎగ్జిక్యూట్ అవుతుంది. ఆ తర్వాతి case స్టేట్‌మెంట్‌లన్నీ దాటి '}' బ్రాకెట్ బయటపడతాము.

ఒక అక్షరాన్ని కూడా scanf ద్వారా char చరరాశికి ఇవ్వవచ్చు. ఇప్పుడు ఒక అక్షరాన్ని ఇవ్వటం ద్వారా మనం switch....case ఎలా పని చేస్తుందో చూద్దాం.

```
#include<stdio.h>
main ()
{
        char C;
        printf("Enter first letter of colour \n");
        scanf("%c", & c);
        switch (C)
        {
                case 'G' : printf("Green");
                break;
                case 'B' : printf ("BLUE");
                break;
                case 'W' : printf("WHITE");
                break;
                case 'R' : printf("RED");
                break'
                case 'Y' : printf("YELLOW");
                break;
        }
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే కింది విధంగా ఫలితం ఉంటుంది.

```
enter the first letter of colour
R
RED.
```

దీన్ని బట్టి స్విచ్ కేస్‌ల ఉపయోగం పూర్తిగా తెలుస్తుంది.

ఒక సంఖ్యను తీసుకుని అది సరి సంఖ్యా లేక బేసి సంఖ్యా అని చెప్పే ప్రోగ్రాంను ఇప్పుడు చూద్దాం. ఏదైనా సంఖ్యను రెండుతో భాగించినప్పుడు శేషం వస్తే అది సరిసంఖ్య, లేనిచో బేసి సంఖ్య అవుతుంది.

```
# include<stdio.h>
main()
{
```

```
    int num1;
    printf("Enter a number \n");
    scanf("%d", &num1);
    if (num1 %2 == 0)
    printf("Even number");
    else
    printf("Odd number");
}
```

ఈ ప్రోగ్రాంను రన్ చేసి ఏమి వస్తుందో చూడండి.

ఇందులో మనం ఒక సంఖ్యను రెండుతో భాగిస్తే వచ్చే శేషం సున్నా అవుతుందేమో చూశాం. సున్నా అయితే సరి సంఖ్య అని, కాకపోతే బేసి సంఖ్య అని ప్రింట్ చేస్తున్నాం.

కండిషనల్ ఆపరేటర్ (Conditional Operator) :

ఇప్పటివరకూ if....else నియమాలను, switch.....ease నియమాలను చూశాం. అయితే చిన్న చిన్న నియమాలను పరిశీలించడానికి కూడా మొత్తం if....else ను వీడనవసరం లేకుండా 'సి' భాష మనకు కండిషనల్ ఆపరేటర్‌ను అందిస్తోంది. సాధారణంగా x>y అయితే x=10 అని, కాకపోతే y=10 అని రాయాలి. దానికి మనం

```
if (x>y)
x = 10;
else
y = 10;
```

అని రాస్తాం. అయితే దీన్ని కండిషనల్ ఆపరేటర్ ఉపయోగించి ఒకే లైన్‌లో రాయవచ్చు.

ఉదాహరణకి

```
(x>y)? x=10 : y=10;
```

అని రాయవచ్చు. పైన రాసిన if.....else స్టేట్‌మెంట్ ఫలితమే వస్తుంది. ఈ ఆపరేటర్ సాధారణ స్వరూపం కింది విధంగా ఉంటుంది.

నియమం? మొదటి ఫలితం : రెండో ఫలితం;

ఇందులో ముందుగా నియమాన్ని రాయాలి తర్వాత ప్రశ్నార్థకం రాసి దాని తర్వాత ఆ నియమం సత్యమైతే ఏం చెయ్యాలో రాయాలి. దాని తర్వాత కోలన్ ఇచ్చి నియమం సత్యం కాకపోతే ఏం చెయ్యాలో రాయాలి.

```
(x>5)? x=10 : x=1;
```

x>5 అయితే x=10 అవుతుంది. కాకపోతే x=1 అవుతుంది. కింది ఉదాహరణ పరిశీలిస్తే కండిషనల్ ఆపరేట
పూర్తిగా అర్థమవుతుంది.

```
#include<stdio.h>
main ()
{
        int num;
        printf("Enter a number \n");
        scanf("%d",&num);
       (num>0)?printf("positive"):printf("negative");
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేయండి.

```
Enter a number
10
Positive
```

అని ప్రింటవుతుంది.

అభ్యాసం

1. నెలల సంఖ్యలు ఎంటర్ చేస్తే ఆ నెల పేరు వచ్చేట్లుగా ప్రోగ్రాం రాయండి.
   ఉదా : 3 అని ఇస్తే March అని, 9 అని ఇస్తే October అని ప్రింట్ కావాలి.
2. రెండు సంఖ్యలను తీసుకుని మొదటిది రెండో సంఖ్యలో సంపూర్ణంగా భాగించబడుతుందా లేదా అని పరీక్షించండి. (శేషం సున్నా కావాలి)
3. రన్ చేస్తున్న సమయంలో 3 సంఖ్యలను తీసుకుని వాటిలో ఏది పెద్దదో నిర్ణయించండి.



అధ్యాయం 7

వలయాలు
(Loops)

ఒక చరరాశి విలువను పెంచాలంటే దానికి కొంత విలువను కలపాలి. అదే విలువను తగ్గించాలంటే కొంత విలువను తీసెయ్యాలి. ఉదాహరణకి x అనే చరరాశి విలువ 4 అనుకోండి. దానికి 6 కలపాలంటే

x = x+6;

అని రాస్తాం.

అప్పుడు x విలువ 10 అవుతుంది. అదే x లోంచి 3 తీసెయ్యాలంటే

x = x–3;

అనిరాస్తాం.

ఈ కూడిక తీసివేత ప్రక్రియలను సి భాష మరింత సులభతరం చేస్తుంది.

దాదాపుగా అన్ని కంప్యూటర్ భాషలలోను (మామూలు గణితంలో కూడా) కలపటానికి, తీసేయడానికి ఇదే పద్ధతి. 'సి' భాషలో ఈ కూడికను

x+ = 6;

అని కూడా రాయవచ్చు.

అంటే x = x+6 అని రాసినా, x+ = 6 అని రాసినా అర్థం ఒకటే - x కు 6 కలపాలని.

అదే తీసివేతకు

x–=3;

అని రాయవచ్చు.

x=x–3;

అనే దీని అర్థం.

ఇదే విధంగా హెచ్చవేత, భాగహారం, శేషం వంటి ప్రక్రియలను చేయవచ్చు.

x = x*5;

అనే ప్రక్రియను

x* = 5;

అని రాయవచ్చు.

x = x/2; అన్నా, x/ = 2; అన్నా అర్థం ఒకటే.

x విలువ 9 అయితే

x = x%4 విలువ 1 అవుతుంది.

దీన్ని కూడా x% = 4 అని రాయవచ్చు.

ఒక చరరాశి విలువకు 1 కలపాలన్నా తీసేయాలన్నా మరో తేలికైన పద్ధతి ఉంది. సాధారణంగా ఒక చరరాశి x క 1 కలపాలంటే

x = x+1 అనో లేదా x+ =1 అనో రాస్తాం. దీనికి బదులుగా

x++ (x పక్కన '+'ను రెండుసార్లు - మధ్యలో ఖాళీ లేకుండా రాయాలి).

ఉదాహరణకు x విలువ 4 అనుకుంటే

x++;

అని రాసిన తర్వాత x విలువ 5 అవుతుంది.

అదే x విలువ 1 తగ్గించాలంటే,

x--;

అని రాయవచ్చు.

ఉదాహరణకు x విలువ 4 అనుకుందాం.

x--;

తర్వాత x విలువ 3 కు తగ్గుతుంది. రెండు గుర్తులకి మధ్య ఖాళీ ఉండకూడదు.

పైన నేర్చుకున్న ప్రక్రియలను ఉపయోగించి ఒక ప్రోగ్రాం చేద్దాం.

```
# include<stdio.h>
main()
{
    int num1, num2, result;
    printf("Please enter a number \n");
    scanf(%d", &num1);
    printf("Please enter another number \n");
    scanf("%d", &num2);
    printf("1. Adding 10 to first number \n");
    num1 + = 10;
    printf("The result is %d \n", num1);
```

```
printf("2. Subtracting 5 from second number \n");
num1 - = 5;
printf("3. Multiplying first numer with 4 \n");
num1* = 4;
printf("The results of above opertations are %d and %d", num2, num1);
printf("4. Dividing first number by 2");
num1/ = 2;
printf("5. Remainder after dividing first number by three \n");
num1% = 3;
printf(6. Increamenting second number \n");
num2++;
printf("result is %d \n", num2);
printf("Decrementing first number \n");
num1--;
printf("result is %d \n", num1);
}
```

ఒక అంకె లేదా సంఖ్య విలువను ఒకటి పెంచటం (Increment) లేదా తగ్గించటానికి (Decrement) ఉపయోగించే ఆపరేటర్లు కేవలం 'సి' భాషలో మాత్రమే ఉన్నాయి. నిజానికి కంప్యూటర్‌లోని అతి తక్కువ స్థాయి భాష (lower level language) అయిన అసెంబ్లీ భాషలో మాత్రమే ఉన్న ఈ అంశాలను 'సి' చాలా సమర్ధంగా తనలో ఇముడ్చుకుంది.

(గమనిక : ఇంతకు ముందు వివరించినట్లు కంప్యూటర్‌కు మెదడులాంటిదైన మైక్రోప్రాసెసర్‌కు కేవలం ఒక భాష మాత్రమే అర్థమవుతుంది. ఇందులో రాసిన కమాండ్స్‌ను మైక్రోప్రాసెసర్ అమలు చేస్తుంది. మనం ఏ ప్రోగ్రామింగ్ భాష రాసిన మైక్రోప్రాసెసర్‌కు అందే సూచనలు మాత్రం అసెంబ్లీ భాషలోనే ఉంటాయి. ఇంతకు ముందు చెప్పిన కంపైలర్లు లేదా ఇంటర్‌ప్రెటర్లు ఈ ప్రోగ్రామింగ్ భాషను అసెంబ్లీ భాషలోకి మార్చడానికి ఉపయోగపడతాయి.)

ఇక అసలు విషయానికి వద్దాం.

ఒక పేరుని పదిసార్లు జపించాలంటే ఏంచేస్తారు? చిన్న జపమాల తీసుకొని ఒక్కోపూస లేదా రుద్రాక్షను జరుపుతూ పేరుని జపిస్తారు. పది అయిపోయిన తర్వాత జపం ఆపేస్తారు. గడియారంలో నిమిషాల ముల్లు ప్రతిగంటకొకసారి మొత్తం వలయాన్ని పూర్తి చేస్తుంది. ఇప్పుడు మీరు గడియారం చూడటం ఆపేస్తారు. మరికొంత సేపటి తర్వాత చూస్తారు. అప్పటికి నిమిషాల ముల్లు ఎన్ని వలయాలు పూర్తి చేసిందని ఎవరైనా అడిగితే మీరు సమాధానం ఎలాచెబుతారు? చాలా తేలిక. గంటల ముల్లును చూసి చెప్పవచ్చు. గంటలముల్లు ఎన్ని గంటలు జరిగితే నిమిషాల ముల్లు అన్ని వలయాలు పూర్తి చేస్తోందన్నమాట. అంటే నిమిషాల ముల్లు ఎన్ని వలయాలను పూర్తి చేసిందో గంటల ముల్లు లెక్కపెడుతోంది. అదే విధంగా సెకన్లముల్లు ఎన్ని వలయాలు పూర్తి చేసిందో నిమిషాల ముల్లు గణిస్తుంది. మరి గంటల ముల్లు ఎన్ని వలయాలు తిరిగిందో ఏది లెక్కపెడుతుంది. ఒకరోజు. గంటల ముల్లు ఒక రోజులో రెండు వలయాలను

పూర్తి చేస్తుంది. అంటే ఎన్ని రోజుల సమయం పూర్తయిందో కనుక్కుంటే గంటల ముల్లు ఎన్ని వలయాలు తిరిగిందో దాని ద్వార నిమిషాలముల్లు ఎన్ని రౌండ్లు వేసిందో, సెకన్ల ముల్లు ఎన్ని చక్కర్లు కొట్టిందో తేలిగ్గా చెప్పవచ్చు. ఇక్కడ ఇంకో విషయం గమనించండి. నిమిషం పూర్తయ్యేసరికి సెకన్లముల్లు మళ్లీ 12 నుంచి మొదలవుతుంది. అదేవిధంగా గంటపూర్తయ్యేసరికి నిమిషాలముల్లు, సెకన్ల ముల్లుకూడా 12 నుంచి మొదలవుతాయి. సగం రోజు పూర్తయ్యేసరికి గంటల ముల్లు, నిమిషాల ముల్లు, సెకన్ల ముల్లూ కూడ 12 నుంచి మొదలవుతాయి. దీన్నిబట్టి పై వలయం ఒకటి జరిగితే వాటి కింద వలయాలన్నీ మొదటి నుంచి ప్రారంభమవుతాయని తెలుస్తోంది. పై విషయాలను దృష్టిలో ఉంచుకుని క్రింది పాఠాన్ని గమనించండి.

ఒక పేరుని 4 సార్లు ప్రింట్ చేయాలి. దీనికి ప్రోగ్రాం రాయాలి. మీరు ఎలారాస్తారు? అదే పేరుని నాలుగు printf స్టేట్‌మెంట్‌లని ఉపయోగించి రాయవచ్చు లేదా Loop ని ఉపయోగించి రాయవచ్చు. ఈ Loop ని ఉపయోగిస్తే ఒకే printf స్టేట్‌మెంట్‌ని మనం నాలుగుసార్లు ఎగ్జిక్యూట్ చేయవచ్చు. Loop (వలయం) అంటే ఏమిటో ఇప్పుడు చూద్దాం.

ప్రోగ్రాంలో చాలాసార్లు ఎగ్జిక్యూట్ అయ్యేకోడ్ లేదా స్టేట్‌మెంట్ల సమూహాన్ని వలయం అంటారు. ఇప్పటి వరకూ మనం సాధారణ, తేలికైన ప్రోగ్రాంలు చూశాం. ఇప్పుడు మనకు 'సి' కొంత అలవాటైంది కాబట్టి ఇక కొంత పెద్ద ప్రోగ్రాంలను రాయటం అలవాటు చేసుకోవచ్చు. అందుకనే ఈ వలయాలను ఉపయోగించి ప్రోగ్రాంలు రాద్దాం. ప్రోగ్రాంలో ఒకసారి రాసిన కోడ్‌ను మనకు కాలసినన్నిసార్లు ఎగ్జిక్యూట్ చేయగలిగేవే ఈ వలయాలు: ఉదాహరణకి

printf("Hello World");

అనే స్టేట్‌మెంట్ ఒకసారి ఎగ్జిక్యూట్ అయితే Hello World అని ఒకసారి ప్రింటవుతుంది. అదే అయిదుసార్లు ఎగ్జిక్యూట్ అయితే అయిదుసార్లు ప్రింటవుతుంది.

num++; అనే స్టేట్‌మెంట్ ఒకసారి ఎగ్జిక్యూట్ అయితే num విలువ 1 పెరుగుతుంది. అదే 5 సార్లు ఎగ్జిక్యూట్ అయితే num విలువకు 5 కలుస్తుంది.

while వలయం

ఈ లూప్‌ను సాధారణంగా ఇలారాస్తారు.

while (షరతు) స్టేట్‌మెంట్‌లు

ఈ క్రింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int i =1;
    while(i<6)
    {
```

```
        printf("Hello \n");
        i++;
      }
    }
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే ఈ క్రింది ఫలితం ప్రింటవుతుంది.

Hello

Hello

Hello

Hello

Hello

ఇది ఏవిధంగా సాధ్యమైంది?

మనం పైన ప్రోగ్రాంలో i విలువ 6 కంటే తక్కువగా ఉన్నంతవరకూ రెండు బ్రాకెట్ల మధ్య స్టేట్‌మెంట్‌లను ఎగ్జిక్యూట్ చేయమని సూచన ఇచ్చాం. i విలువ 1 అని పైన డిక్లరేషన్‌లో ఇచ్చాం. కాబట్టి i విలువ 6 కంటే తక్కువే అందువల్ల printf స్టేట్‌మెంట్ ఎగ్జిక్యూట్ అవుతుంది. Hello అని ప్రింటవుతుంది. ఇప్పుడు printf కింద i++; అని మరోస్టేట్‌మెంట్ ఇచ్చాం. దాంతో i విలువ 1 పెరిగి 2కి చేరుతుంది. ఇప్పుడు while పక్కన ఉన్న షరతు మరోసారి పరీక్షించబడుతుంది i విలువ ఇంకా 6 కంటే తక్కువే.

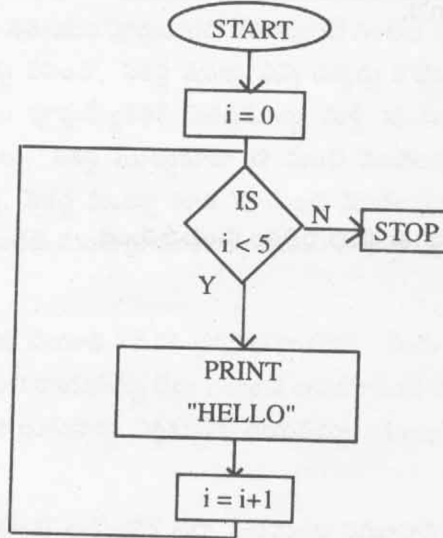
కాబట్టి ఈ రెండు స్టేట్‌మెంట్‌లు మళ్ళీ ఎగ్జిక్యూట్ అవుతాయి.

Hello మరోసారి ప్రింటవుతుంది.

విలువ 1 పెరుగుతుంది. ఇప్పుడు i విలువ 3 అవుతుంది.

షరతును మరోసారి పరిశీలిస్తుంది. i విలువ 6 కంటే తక్కువే కాబట్టి మరోసారి Hello ప్రింటవుతుంది. ఈ సారి i విలువ 1 పెరిగి 4 కి చేరుతుంది. i ఇంకా 6 కంటే తక్కువే. అందువల్ల మరోసారి Hello ప్రింటవుతుంది. i విలువ 5కి చేరుతుంది. ఇప్పుడు కూడా షరతు నెరవేరలేదు కాబట్టి Hello మరోసారి ప్రింటవుతుంది. ఈ సారి i విలువ 1 పెరిగి 6 అవుతుంది. ఇప్పుడు i<6 అనే షరతులను పరిశీలిస్తే ఈషరతు అసత్యమవుతుంది ఎందుకంటే ఇప్పుడు i విలువ 6కి సమానం కాబట్టి, దీంతో while బ్రాకెట్ల నుంచి బయటపడి ప్రోగ్రాం ముగుస్తుంది. ఫలితంగా Hello అనే పదం 5 సార్లు ప్రింటయింది.

ఈ వలయపు ప్రోగ్రామ్‌ని మరింత తేలికగా అర్ధం చేసుకోవడానికి బొమ్మ రూపంలో చూద్దామా! ఈ వలయానికి సంబంధించి Flow chart ఈ కింది విధంగా ఉంటుంది.



ఈ ఫ్లో ఛార్ట్‌ను బట్టి while వలయాన్ని అర్థం చేసుకోవచ్చు.

ఇప్పుడు ఈ కింది అంశాలను పరిశీలించండి.

```
int i=6;
while(i<6)
{
    Printf("Hello");
    i++;
}
```

ఇందులో Hello ఎన్నిసార్లు ప్రింటవుతుంది?

ఒక్కసారికూడా ప్రింట్‌కాదు, కారణమేమిటి?

డిక్లేర్ చేసే సమయంలోనే i విలువ 6 అని ఇచ్చాం. ఇక while పక్కన షరతును పరిశీలించేటప్పుడు i విలువ 6కి సమానమని తెలుస్తుంది. దాంతో షరతు అసత్యమవుతుంది. అందువల్ల కనీసం ఒక సారి కూడా ప్రింట్ కాదు.

ఈ కింది అంశాన్ని చూడండి.

```
int i= 3;
while(i<6)
{
    printf("Hello");
    i++;
}
```

ఇప్పుడు Hello అని మూడు సార్లు ప్రింటవుతుంది. అంటే ఈ వలయం మూడు సార్లు ఎగ్జిక్యూట్ అయ్యేసరికి i విలువ 6కి చేరుతుంది. దాంతో వలయం ఆగిపోతుంది. ఇంకోముఖ్యమైన విషయాన్ని పరిశీలించండి.

```
i = 1;
while(i<6)
{
    printf("Hello");
}
```

ఇప్పుడు Hello అని ఎన్నిసార్లు ప్రింటవుతుంది? సమాధానం చెప్పేముందు పైప్రోగ్రాంను మరోసారి పరిశీలించండి. ఇప్పుడు చెప్పండి.

సమధానం....... కొన్ని వందలు, వేలు, లక్షలు, కోట్లు అనంతంగా అలా ప్రింటవుతూనే ఉంటుంది. మీరు ఆపనంతవరకూ, మీ కంప్యూటర్‌కి కరెంట్ వస్తునంతసేపూ ప్రింటవుతుంది. దీనిని అనంతవలయం (Infinite loop) అంటారు.

ఎన్ని వలయాలు పూర్తయినా i విలువ 1 గానే ఉంటుంది. ఎందుకంటే మనం i విలువని పెంచటంలేదు. అందువల్ల షరతు సంపూర్తి కాదు. దాంతో అనంతవలయం ఏర్పడుతుంది. ఇది ఏర్పడకుండా ప్రోగ్రాంలో జాగ్రత్తలు తీసుకోవాలి.

ఇప్పుడు ఇంకో ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main()
{
    int i;
    i=1;
    while(i<21)
    {
        printf("%d \n", i);
        i++;
    }
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే 1 నుంచి 20 వరకూ సంఖ్యలు ప్రింటవుతాయి.

i విలువలను మనం ప్రింట్ చేస్తున్నాం. i విలువ 21 చేరేవరకూ అన్ని అంకెలూ ప్రింటవుతాయి. ఈ ప్రోగ్రాంలోని printf స్టేట్‌మెంట్‌లో i స్థానంలో i+2ని ఉంచి ఈ ప్రోగ్రాం మళ్లీరన్ చేయండి. ఇప్పుడు 2, 4, 6, 8 .... 22 వరకూ అన్నీ

సరి సంఖ్యలే ప్రింటవుతాయి. i ని ప్రింట్ చేసే ప్రతిసారి i కి 2 కలుపుతున్నాం. అందువల్ల i పెరిగేకొద్దీ ప్రింటయ్యే సంఖ్య విలువకూడా పెరుగుతుంది. ఇదే మరో ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int i;
    i=20;
    while(i>0)
    {
        Printf("%d \n", i)
        i--;
    }
}
```

ఇప్పుడు సంఖ్యలు 20, 19 .... 1 వరకూ ప్రింటవుతాయి. ఇక్కడ i విలువ 0 కి చేరేవరకూ వలయం ఉంటుంది.

మరో ప్రోగ్రాం చూద్దాం. ఇక్కడ మనం ఒక విలువను తీసుకుని దాన్ని ఘనం (square) చేస్తున్నాం. అయిదు విలువలను తీసుకుని చూద్దాం.

```
#include<stdio.h>
main()
{
    int num, i=1;
    while(i<6)
    {
        printf("Enter a number \n");
        scanf("%d", & num);
        printf("The square of the number %d is %d", num, num*num);
        i++;
    }
}
```

దీన్ని ఎగ్జిక్యూట్ చేస్తే

Enter a number

5 - అని మనం ఎంటర్ చేయాలి. అప్పుడు...

The square of 5 is 25

అని వస్తుంది. ఈ విధంగా 5 సంఖ్యలను అడిగి వాటి ఘనాలను ఇస్తుంది. ఇదేవిధంగా ఒక వృత్తానికి వ్యాసార్ధం రేడియస్‌ను తీసుకుని ఆ వృత్తం వైశాల్యం చెప్పవచ్చు. ఎన్ని వృత్తాల వైశాల్యాలనయినా కనుక్కోవచ్చు.

వృత్తాల వైశాల్యాన్ని కనుకోడానికి ఒక ప్రోగ్రాం రాయండి.

(గమనిక వృత్త వైశాల్యం = 3.14 *r*r) ఇక్కడ

r అంటే రేడియస్ దీన్ని scanf ద్వారా తీసుకోవాలి.

వృత్త వైశాల్యం తీసుకునేటప్పుడు %d కి బదులుగా %f వాడండి ఎందుకంటే కి 3.14 తో హెచ్చివేస్తే ఫలితాల్లో దశాంశస్థానం ఉంటుంది.)

ఇటువంటివే కొన్ని ప్రోగ్రాంలను ఈ పాఠం చివర అభ్యాసంగా ఇచ్చాం. వాటికి ప్రోగ్రాంలను రాయండి.

Do.... while వలయం

- ఇప్పటి వరకూ మనం while వలయాన్ని చూశాం. అందులో ముందుగా షరతును పరీక్షించి ఆ షరతు సత్యమైతేనే వలయం లోపలి స్టేట్‌మెంట్‌లు ఎగ్జిక్యూట్ అవుతాయి. ఇదే ఆ షరతు సత్యమైనంత సేపు మన వలయం ఎగ్జిక్యూట్ అవుతుంది. ఇక్కడ మరో రకమైన వలయాన్ని చూద్దాం.

```
Do
{
    statements;
}   while(expression);
```

ఈ వలయం ముందుగా Do తో ప్రారంభమవుతుంది. బ్రాకెట్ల మధ్యలో స్టేట్‌మెంట్‌లుంటాయి. చివర while ఉండి షరతు ఉంటుంది. ఈ కింది ఉదాహరణను పరిశీలించండి.

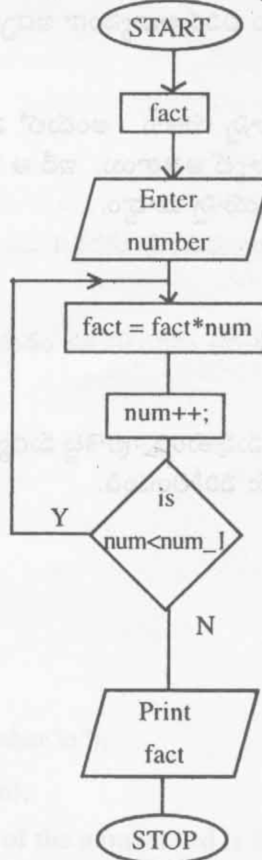
```
#include<stdio.h>
main()
{
    int num1,num2;
    long fact = 1;
    printf("Enter a number");
        scanf("%d", &num");
    num2 =1;
Do
{
    fact = fact* num2;
    num2 ++;
}while(num2<=num1);
```

```
printf("The factorial of %d is %ld", num1,fact);
}
```

ఇందులో మనం వలయం పూర్తయ్యాక షరతును పరీక్షిస్తున్నాం. అంటే ఆ వలయాన్ని ఒకసారి దాటాకే షరతును చూస్తున్నాం. while వలయానికి, దీనికి ఉన్న తేడా అదే. ముందు చూసిన while వలయంలో ముందు షరతును పరీక్షిస్తాం. షరతు అసత్యమైతే వలయాన్ని ఒకసారి కూడ ఎగ్జిక్యూట్ చేయము. అదే Do....while వలయంలో షరతును పరీక్షించకుండానే వలయాన్ని ఒక సారి ఎగ్జిక్యూట్ చేస్తాం. షరతు అసత్యమైనా సరే వలయం ఒక సారి ఎగ్జిక్యూట్ అవుతుంది.

పై ఉదాహరణలో ఒక సంఖ్యను తీసుకుని factorial ను లెక్కకట్టే ప్రోగ్రాం చూశాం. int విలువ 32,767 లోపలే ఉండాలి. కాని factorial విలువలు చాలా ఎక్కువుగా ఉంటాయి. అందువల్ల దీన్ని long అని తీసుకున్నాం.

Do... while వలయానికి పట్టిక ఏ విధంగా ఉంటుందో ఇప్పుడు చూద్దాం.



ఇప్పుడు మరో ప్రత్యేక ప్రోగ్రాం చూద్దాం. దీన్ని జాగ్రత్తగా పరిశీలించి ఎలా చేశామో గ్రహించండి. ఇందులో ఒక ధన సంఖ్య (Positive number) ఎంటర్ చేస్తే POSITIVE అని, ఋణ సంఖ్య (Negative number) ఎంటర్ చేస్తే NEGETIVE అనివస్తుంది. 0 ఎంటర్ చేస్తే మరల ఎంటర్ ఎ నంబర్ అని వస్తుంది.

```
#include<stdio.h>
main
```

```
{
    int num = 1;
do
{
    printf("Enter a number \n");
    scanf ("%d", &num);
    if(num>0)
    printf("POSITIVE");
    else if(num<0)
    printf("NEGETIVE");
    }while(num==0)
    printf("PROGRAME WILL CONTINUE");
}
```

ఇప్పుడు దీన్ని ఎగ్జిక్యూట్ చేయండి. 0 ఇచ్చేదాకా ఈ ప్రోగ్రాం వస్తూనే ఉంటుంది.

num విలువ 0 కి సమానం కానంతవరకూ వలయంలో తిరుగుతూ ఉండమని ఆదేశం ఇచ్చాంకదా? ఈ ప్రోగ్రాం ద్వారా మనకు Do.......while వలయం ఉపయోగం తెలుస్తుంది. దీన్నే while ఉపయోగించి చేసి చూడండి తేడా మీకే అర్థమవుతుంది.

For వలయం– 'సి' భాషలో చాలా శక్తిమంతమైనది ఈ for వలయం మిగిలిన వలయాలకంటే ఇదే ఎక్కువ ఉపయోగంలో ఉంది.

పైన పేర్కొన్న while వలయాన్ని ఉపయోగించి ఒక ప్రోగ్రాం చేశాం దీన్ని మళ్లీ చూద్దాం.

```
#include<stdio.h>
main()
{
    int i=1;
    while(i<10)
    {
        printf("%d \n", i);
        i++;
    }
}
```

ఈ ప్రోగ్రాంలో మూడు ముఖ్యమైన అంశాలను గమనించండి. మొదటిది i అనే చరరాశికి ఒక ప్రారంభవిలువను ఇచ్చాం. రెండవది, i లో విలువ 10 కి చేరిందా లేదా అని పరీక్షిస్తున్నాం. మూడవది i విలువని ఒక్కొక్కటిగా పెంచుతున్నాం. వలయాలు ఉపయోగించే దాదాపు అన్ని ప్రోగ్రాముల్లోను ఈ మూడు అంశాలు ఉన్నాయి గమనించండి. అయితే for వలయంలో ఈ మూడు అంశాలు ఒకే చోట వస్తాయి. దీని సాధారణ స్వరూపం ఈ కింది విధంగా ఉంటుంది.

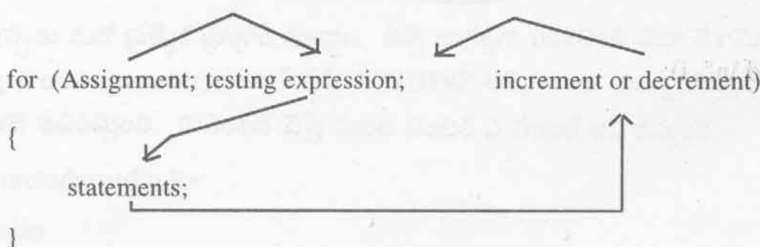
```
for (initialisation;expression;increment or decrement)
{
    statements;
}
```

దీన్ని ఉపయోగించి ఒక ప్రోగ్రాం చేద్దాం.

```
#include<stdio.h>
main()
{
    int i;
    for(i=1;i<21;i++)
    {
        printf("%d", i);
    }
}
```

పైన పేర్కొన్న మూడు ప్రధాన అంశాలను ఈ ప్రోగ్రాంలో ఒకే for పక్కనే రాశాం. ఇందులో ముందుగా i కి ఒక విలువ ఇస్తున్నాం. తర్వాత షరతును పరీక్షిస్తున్నాం. షరతు సత్యమైతే వలయాన్ని ఎగ్జిక్యూట్ చేస్తాం. తర్వాత i విలువ 1 పెంచుదాం. మళ్లీ షరతును పరీక్షించి వలయాన్ని ఎగ్జిక్యూట్ చేస్తాం. for వలయంలో ముఖ్యమైన అంశాలు.

1. చరరాశికి విలువను ఇవ్వటం, దాని షరతును పరీక్షించటం, విలువలను పెంచటం ఒకే చోట ఉంటాయి. వీటి మధ్య ఒక సెమికోలన్ తప్పనిసరిగా ఉండాలి.
2. ముందుగా చరరాశికి విలువ ఇవ్వటం (Assignment) తర్వాత షరతును పరీక్షించటం జరుగుతాయి. షరతు సత్యమైతే వలయం ఎగ్జిక్యూట్ అవుతుంది. షరతు అసత్యమైతే వలయం ఎగ్జిక్యూట్ కాదు.
3. వలయం ఒకసారి ఎగ్జిక్యూట్ అయ్యాక అప్పుడు చరరాశి విలువ పెరగటం లేదా తగ్గటం జరుగుతాయి. మళ్లీ షరతును పరీక్షించబడుతుంది. ఈ వలయం ఎగ్జిక్యూషన్ ఈ కింది విధంగా ఉంటుంది.



4. చరరాశికి విలువను ఇచ్చే స్టేట్మెంట్ ఒకసారి మాత్రమే ఎగ్జిక్యూట్ అవుతుంది.

5. ప్రతిసారి చరరాశి విలువ పెరిగాకే షరతు పరీక్షించబడుతుంది. (మొదటి సారి తప్ప).

6. while వలయంలో మూడు స్టేట్మెంట్లలో చేసిన పనిని ఇక్కడ ఒకే స్టేట్ మెంట్లో చేస్తున్నాం గమనించండి.

ఈ కింది ఉదాహరణను పరిశీలించండి.

```
#include<stdio.h>
main()
{
    int i, sum=0;
    for(i=0; i<=100; i++)
    {
        sum=sum+i;
    }
    printf("The sum of integers from 0 to100 is %d", sum);
}
```

ఇక్కడ మనం 1 నుంచి 100 వరకూ ఉన్న సంఖ్యల మొత్తాన్ని కనుగొంటున్నాం. ప్రోగ్రాం చాలా తేలిగ్గా ఉంది కదూ.

ఇదే రెండు సంఖ్యలని రన్ టైమ్లో తీసుకొని వాటి మధ్య ఉన్న సంఖ్యల మొత్తాన్ని కనుగొనవచ్చు.

రన్టైమ్ (Run time): ఒక ప్రోగ్రాంను ఎగ్జిక్యూట్ చెయ్యటం మొదలు పెట్టాక ప్రతి scanf స్టేట్మెంట్ వద్ద ప్రోగ్రాం ఆగుతుంది. మనం కీబోర్డు ద్వారా విలువలను ఎంటర్ చేశాక ప్రోగ్రాం తిరిగి ప్రారంభమవుతుంది. ఈ విధంగా ప్రోగ్రాం రన్ కావడం మొదలయ్యాక scanf లేదా ఇతర ఇన్పుట్ స్టేట్మెంట్ల ద్వారా చరరాశులకు విలువలను ఇవ్వటాన్ని రన్టైమ్లో విలువలను ఇవ్వటం అంటారు.

```
#include<stdio.h>
main()
{
    int small, big, sum=0,i;
    printf("Enter_small number first");
    scanf("%d", &small);
    printf("Enter big number");
    scanf("%d", &big);
    for(i=small; i<=big; i++)
```

```
        {
            sum=sum+i;
        }
    print("The sum of number between %d and %d , %d", small big, sum),
}
```

ఈ ప్రోగ్రాంను రన్‌చేసి చూడండి.

```
Enter small number first
10
Enter big number
15
The sum of numbers between 10 and 15 is
75
```

దీంతో మీరు ఇప్పుడు 10 సంఖ్యలను తీసుకుని వాటిలో అతి పెద్ద సంఖ్యను గుర్తించే ప్రోగ్రాం చేద్దాం.

ఇది చాలా తేలిక ముందుగా ఒక చరరాశిని తీసుకుని దానికి ఒక విలువను ఇద్దాం. తర్వాత విలువలు తీసుకుంటూ ముందున్న చరరాశి విలువతో పోలుస్తాం. ఒక వేళ చరరాశి విలువ కొత్త విలువకన్నా తక్కువైతే ఆ కొత్త విలువను మన చరరాశికి ఇవ్వాలి. కొత్త విలువ చరరాశికన్నా తక్కువతే అప్పుడు ఏమి చేయక్కరలేదు. ఈ కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int max, num,i;
    max=0;
    for(i=1; i<=10; i++)
    {
        printf("Enter a number \n");
        scanf("%d", & num);
        if (max<num)
        {
            max=num;
        }
```

```
    }
    printf("The maximum of 10 numbers is %d", max);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి చూడండి.

if వద్ద మనం ఒక షరతును పరీక్షిస్తున్నాం. అందులోనే మనం చరరాశి max లో విలువను, num లో విలువను పోలుస్తున్నాం. max లో విలువ num లోని విలువ కన్నా తక్కువైతే మనం num లోని విలువను max కు ఇస్తున్నాం. ఇందువల్ల max లో ఎప్పుడూ అత్యధిక విలువ మాత్రమే ఉంటుంది.

పదిమంది విద్యార్థుల మార్కులను input గా తీసుకుని వారి మార్కుల సగటును కనుగొనే ప్రోగ్రాంను ఇప్పుడు చూద్దాం.

```
#include<stdio.h>
main ()
{
        int sum, mark, i;     sum = 0;
        float avg;
        for (i=0, i<=9; i++)
        {
                print f("Enter mark of student ro:%d", i+1);
                scan f("%d", & mark)
                sum = sum + mark
        }
        avg = (float) sum/10;
        printf("the average is %f\n", avg);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

ఇదే విధంగా వలయాన్ని ఉపయోగించి మరో ప్రోగ్రాం చేద్దాం. ఫిబనాక్కీ సిరీస్ అన్న సిరీస్‌లో సంఖ్యలు ఈ కింది విధంగా ఉంటాయి.

0 1 1 2 3 5 8 13 21 34 ......

ఈ సిరీస్‌ను జాగ్రత్తగా పరిశీలించండి. ఇందులో మూడో సంఖ్య నుంచి ప్రతి సంఖ్య అంతకు ముందు రెండు సంఖ్యల మొత్తానికి సమానం. అంటే ఒకటి, రెండు సంఖ్యలను కలిపితే మూడో సంఖ్య వస్తుంది. రెండు, మూడో

సంఖ్యలను కలిపితే నాలుగో సంఖ్య వస్తుంది. ఇదే విధంగా ఎన్ని సంఖ్యలనయినా తెలుసుకోవచ్చు. ఈ ఫిబనాక్కీ సిరీస్‌లో మొదటి పది సంఖ్యలను తెలుసుకునే ప్రోగ్రాంను ఇప్పుడు చూద్దాం.

```
#include<stdio.h>
main ()
{
        int num_1, num_2, num_3, i;
        /* initial values of num_1 and num_2 are 0 and 1*1
        num_1 = 0;
        num_2 = 1;
        /* print first two numbes */
        printf("%d", num_1);
        printf("%d", num_2);
        for (i = 2; i<=109; i++)
        {
                num_3 = num_2 + num_1;
                /*print third number anmd onwords */
                printf("%d", num_3);
                num_1 = num_2;
                num_2 = num_3;
        }
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

0 1 1 2 3 5 8 13 21 34

అని ప్రింటవుతుంది.

ఒక చిన్న తమాషా ప్రోగ్రాం చూద్దాం. ఈ ప్రోగ్రాంను రన్ చేసిన తర్వాత మీ ఫ్రెండ్‌ని ఒక సంఖ్య మనసులో అనుకోమనండి. ఆ సంఖ్యను మీరు ప్రోగ్రాంలో చెప్పినట్లుగా మార్పులు చేసి సమాధానం టైప్ చేయమనండి. మీ ఫ్రెండ్ కోరుకున్న సంఖ్య అక్కడ ప్రత్యక్షమవుతుంది.

```
#include<stdio.h>
main ()
{
```

```
        int num;
        printf("please imagine a number \n");
        printf("Add 5 to that number \n");
        printf("multiply the result by 3 \n");
        printf("subtract 10 from that number \n");
        printf("Add 13 to that result \n");
        printf(Now enter the final result \n");
        scanf("%d", & num);
        /* Now we reverse the above process */
        num = num_13;
        num = num+10;
        num = num+3;
        num = num_5;
        printf("The number you imagined is %d\n"_num)
}
```

ఈ ప్రోగ్రాంను మీరే ఎగ్జిక్యూట్ చేసి చూడండి. ఫలితం తమాషాగా ఉంటుంది.

ఇప్పుడు రన్‌టైమ్‌లో ఒక సంఖ్యను తీసుకుని దాని ఎక్కం (Table) ప్రింట్ చేసే ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main ()
{
        int x, i;
        printf("Enter the number \n");
        scanf ("%d", & x);
        for (i = 1; i<=10; i++)
        {
                printf ("%d x %d = %d\n", x, i, x*i);
        }       printf("/n");
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే ఫలితం

Enter the number

5

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

అని వస్తుంది.

ఒక సంఖ్యను తీసుకుని దాన్ని తిరగదిప్పి ప్రింట్ చేసే ప్రోగ్రాంను పరిశీలిద్దాం.

```
#include<stdio.h>
main ()
{
        int i, num, reverse;
        printf("Enter a two digit number \n");
        scan f ("%d", & num);
        while (num>0)
        {
                reverse = num+i*10;
                i = num % 10;
                num = num/10;
        }
        printf("The reverse of number is %d", reverse)
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

వలయం లోపల వలయం (Nested Loops)

పాఠం మొదట్లో గడియారం ఉదాహరణ చూశాం కదా అందులో ప్రతినిమిషానికొకసారి ఒక సెకన్ల ముల్లు 60 సెకన్లు పూర్తి చేస్తుంది. మూడు నిమిషాలను మనం ఎలా చూపిస్తాం. దీనికి ఒక చిన్న ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main()
{
    int min, sec;
    for(min = 1; min<=3; min ++)
    {
        printf(" minute %d \n", min);
        for(sec=1;sec<=60; sec++)
        {
            printf("min %d, sec%d\n", min,sec);
        }
    }
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే కింది ఫలితం వస్తుంది.

```
min 1 sec 1
min 1 sec 2
...............
..................
...............
min1 sec 60
min2 sec 1
min2 sec 2
min 2 sec 3
...............
...............
.............
min3 sec1
```

min3 sec2

....

....

....

min3 sec60

పైప్రోగ్రాంలో మొదటి వలయంలో minute ఉంది. అది ఒకసారి ఎగ్జిక్యూట్ కావలంటే లోపల ఉన్న చిన్న వలయం 60 సార్లు ఎగ్జిక్యూట్ కావాలి. అందువల్ల పై వలయం ఎన్నిసార్లు ఎగ్జిక్యూట్ అయితే లోపలి వలయం ఒక్కోసారికి 60 సార్లు ఎగ్జిక్యూట్ అవుతుంది. దీన్ని ఒకటి రెండు సార్లు చూడండి బాగా అర్థమవుతుంది.

మెనూద్వారా నడిచే (Menu driven) ప్రోగ్రాం

మనం నాలుగు పనులు ఒక వరసలో ఉంచి ఏ సంఖ్య ఇన్‌పుట్ చేస్తే ఆ ప్రోగ్రామే ఈ మెనూ ద్వారానడిచే ప్రోగ్రాం. ఈ కింది ఉదాహరణను గమనించండి.

```
#include<stdio.h>
main()
{
    int choice, num1, num2;
    while (choice!=5)
    {
        printf(" ADD \n");
        printf("2. SUBSTRACT \n");
        printf ("3 MULTIPLY \n");
        printf("4 DIVIDE \n");
        printf("5 EXIT");
        printf("Enter your choice \n");
        scanf("%d", &choice);
        printf("Enter two numbers \n");
        scanf("%d", &num);
        scanf("%d", &num2);
        switch(choice)
        {
```

```
case 1: printf("The sum is %d", num1+num2);
break;
case2: printf("the differnece is %d ". num1–num2);
break
case3: printf("The product is %d", num1*num2);
break;
case4: printf("the division is %d", num/num2);
}
}
}
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే ఈ కింది ఫలితం వస్తుంది.

1. ADD
2. SUBSTRACT
3. MULTIPLY
4. DIVIDE
5. EXIT

Enter your choice ---- ఇక్కడ 5 అని ఇస్తే ప్రోగ్రాం బయటికి వస్తారు.

ఈ ప్రోగ్రాంలను ఒకటి రెండుసార్లు చేస్తే మీకు బాగా అర్థమవుతుంది. ఇటువంటివే మరొకిన్న ప్రోగ్రాంలు చేయండి.

అభ్యాసం

1. ఈ కింది విధానంలో సంఖ్యలు ప్రింటయ్యే ప్రోగ్రాం రాయండి.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

(కొంచం కష్టమే అయినా ప్రయత్నించండి).

(క్లూ : ఇందులో రెండు వలయాలు ఒకదానిలోపల ఇంకొటి ఉండాలి)

2. ఒక సంఖ్యను రన్‌టైమ్‌లో తీసుకొని అది ప్రధాన సంఖ్యా? కాదా పరిశీలించండి.

(ప్రధాన సంఖ్య అంటే అది తన చేత లేదా 1 చేత మాత్రమే భాగించవబడుతుంది మిగిలిన ఏ సంఖ్యల చేత అది భాగించ బడదు. ఉదా:- 2, 3, 5, 7, 11, 13, 17, 19, 23 ....)

దీన్ని కూడా ప్రయత్నించండి.

3. కొన్ని సంఖ్యలను రన్‌టైమ్‌లో తీసుకుని వాటి సగటును కనుక్కునే ప్రోగ్రాం రాయండి. ఎన్ని సంఖ్యలను తీసుకోవాలో కూడా రన్‌టైమ్‌లోనే తీసుకోండి.

4. abcd అన్న నాలుగు అక్షరాలు ఈ కింది విధంగా ప్రింటయ్యే ప్రోగ్రాం రాయండి.

abcd
abdc
acbd
acdb
adbc
adcb



అధ్యాయం 8

శ్రేణులు (Arrays)

ఒక విద్యార్థిని లెక్కల పరీక్షలో వచ్చిన మార్కులను నిలువ చేయాలంటే

```
int maths;
```

అని ఒక చరరాశిని డిక్లేర్ చేసుకుని అందులో విలువను నిలువ ఉంచుకుంటాం. అదే వందమంది విద్యార్థుల మార్కులను నిలవ చేయాలంటే ఏంచేస్తాం! వంద చరరాశులను డిక్లేర్ చేయాల్సిందేనా? అప్పుడు ప్రోగ్రామంతా చరరాశుల డిక్లరేషన్‌తోనే సరిపోతుంది. దీనిని తప్పించుకునేందుకు 'సి' భాషశ్రేణుల (Arrays) ను ప్రవేశపెట్టింది.

ఒక తరగతిలో వందమంది విద్యార్థులుంటారు. వారందరి పేర్లను గుర్తు పెట్టుకోవటం కష్టం. అందుకని హాజరు పట్టీ (Attendence Sheet) లో ప్రతి విద్యార్థికి ఒక్కో నంబరును ఇస్తారు. మార్కులు వేసేటప్పుడు కూడా 1వ నంబరు విద్యార్థికి 70 మార్కులు, 2వ నంబరు విద్యార్థికి 80 మార్కులు అని ఇస్తారు. ఇక విద్యార్థులందరికీ ఉమ్మడిగా ఒకటే పేరుంటుంది. ఇదే తరగతి పేరు. ఎనిమిదోతరగతిలో 1వ విద్యార్థి, ఎనిమిదో తరగతిలో 2వ విద్యార్థి .... ఈ విధంగా పేర్లుంటాయి. మొత్తమ్మీద 8(100) అని వేస్తే ఎనిమిదో తరగతిలో వందమంది విద్యార్థులున్నారని తెలుసుకోవటం కష్టం కాదు. శంకర్ పేరుతో ముగ్గురు విద్యార్థులున్నారనుకుందాం అప్పుడు శంకర్(1), శంకర్(2), శంకర్(3) అని వారిని పిలవచ్చుకదా!

ఇదేవిధంగా వందమందికి లెక్కల్లో మార్కులను సూచించడానికి మనం maths(1), maths(2) .... maths(100) అని రాయవచ్చుకదా. ఇప్పుడు వారికి మార్కులను కూడా కేటాయించవచ్చు.

```
maths(1) = 70
maths(2) = 80
....................
....................
....................
maths(100) = 76
```

ఈ విధంగా వందమంది విద్యార్థులకు మార్కులు కేటాయించవచ్చు. వీరిలో 10వ విద్యార్థికి ఎన్ని మార్కులు వచ్చాయంటే maths(10) ని చూడవచ్చు. అన్ని ప్రోగ్రామింగ్ భాషల్లోను ఈ విధంగా ఒకే పేరుని ఉపయోగించి చాలా విలువలను నిలవ ఉంచుకునే వీలుంది. వీటినే శ్రేణులు (Arrays) అంటారు. చరరాశి పక్కన బ్రాకెట్లో మనం ఇచ్చే

విలువ మనం ఎన్నో విద్యార్థిని లేదా ఎన్నో చరరాశికి సూచిస్తున్నామో తెలియజేస్తుంది కాబట్టి ఆ విలువని సూచీ (Index) అంటారు. ఇప్పుడు 'సి'లో శ్రేణులను ఏవిధంగా ఉపయోగిస్తారో చూద్దాం.

శ్రేణిని డిక్లేర్ చేయటం: శ్రేణిని ఒకే పేరుతో సూచిస్తాం. అదేవిధంగా శ్రేణిలోని అన్ని విలువలు ఒకే విధమైన డేటాను కలిగి ఉంటాయి. మామూలు చరరాశులలాగానే శ్రేణుల చరరాశులను కూడా డిక్లేర్ చేయాలి. వీటిని ఈ కింది విధంగా డిక్లేర్ చేయవచ్చు.

```
int mark[100];
float temp[20];
char name[30];
```

మొదటి శ్రేణిలో వంద int విలువలను రెండో శ్రేణిలో 20 float విలువలను మూడో శ్రేణులో 30 char విలువలను నిలవ ఉంచవచ్చు. ఇక్కడ మనం ఎన్ని విలువలను తీసుకోదలచుకున్నామో ముందే చెప్పాలి. అందువల్ల శ్రేణి పరిమాణాన్ని ముందే డిక్లేర్ చెయ్యాలి. ఇక name అనే చరరాశిలో 30 విలువలను మాత్రమే నిలవ ఉంచే వీలుంటుంది. 31వ విలువను నిలవ ఉంచే ప్రయత్నం చేస్తే error వస్తుంది.

సూచీ ఉపయోగించి మనం విలువలను సూచించవచ్చని ముందే తెలుసుకున్నాం. 'సి' భాషలో సాధారణంగా సూచి '0' తో మొదలవుతుంది. కాబట్టి సూచీ విలువ శ్రేణి పరిమాణాల కంటే ఒకటి తక్కువగా ఉంటుంది. అంటే

name[30] అనే శ్రేణిలో సూచి 0 వద్ద మొదలై 29 వద్ద ముగుస్తుంది. అదే mark[100] లో సూచీ 0 వద్ద మొదలై 99 వద్ద ముగుస్తుంది.

ఇప్పుడు వీటికి విలువలను ఇవ్వాలంటే ఈ కింది విధంగా ఇవ్వవచ్చు.

```
mark[0] = 70;
mark[1] = 40;
mark[2] = 60;
....................
..................
mark [99] = 85;
```

ఇక name అనే చరరాశికి విలువలు ఇవ్వాలంటే

```
name[0] = 'R'
name [1] = 'k';
..................
..................
name[29] = 'Z';
```

ఈ విధంగా ఇవ్వవచ్చు. ఈ శ్రేణులను మిగిలిన చరరాశులలాగానే ఉపయోగించవచ్చు. ఉదాహరణకి

mark[20] = mark[10]+mark[5]

21వ చరరాశి (ఇక్కడ 21వ చరరాశి సూచీ 20 అవుతుంది. ఎందుకంటే సూచీ విలువ 0 నుంచి ప్రారంభమవుతుంది. కాబట్టి) విలువ 11వ, 6వ చరరాశుల విలువల మొత్తానికి సమానం.

ఇప్పుడు శ్రేణిని ఉపయోగించి ఒక చిన్న ప్రోగ్రాం చేసి చూద్దాం.

```
#include<stdio.h>
main()
{
    int mark[4], i;
    for (i=0;i<=3; i++)
    {
      printf("Enter the mark \n");
      scanf("%d", &mark[i]);
    }
      for(i=0; i<4; i++)
    {
    printf("%d \n", mark[i]);
    }
}
```

ఈ ప్రోగ్రాంలో మనం సూచీ ఉండాల్సిన స్థానంలో i అనే చరరాశిని ఉపయోగించాం. వలయంలో i విలువ 0, 1, 2, 3 లను సంతరించుకుంటుంది. దాని ప్రకారమే mark లో సూచీ విలువ మారి నాలుగు చరాశులకు విలువలను scanf ద్వారా తీసుకునే అవకాశం వచ్చింది. రెండో వలయంలో ఆ చరరాశుల విలువలను ప్రింట్ చేస్తున్నాం. ఈ విధంగా మనం విలువలను తీసుకోవచ్చు.

ఇప్పుడు ఈ శ్రేణులను పూర్తిగా అర్థం చేసుకుందాం. ఒక చరరాశికి ఒకటికన్నా ఎక్కువ విలువలను నిలువ చేసుకునే సామర్థ్యాన్ని ఇవ్వటం ద్వారా శ్రేణిని సృష్టించవచ్చు. మామూలుగా చరరాశిగా కంప్యూటర్ మెమొరీలో ఈ కింది విధంగా సూచిస్తారు.

x ఆ చరరాశి విలువను మనం చతురస్రంలో రాస్తాం. ఉదాహరణకి

x = 100 అయితే

| 100 | అని చూపవచ్చు.

అదే శ్రేణిగా డిక్లేర్ చేసినప్పుడు x[5] అనే చరరాశిని ఈ విధంగా సూచించవచ్చు.

x | 0 | 1 | 2 | 3 | 4 | ఇప్పుడు x[5] అనే చరరాశిలో 5 విలువలను నిలువ చేసుకోవచ్చు. అంటే x కి 10, 2, 4, 3, 20 అనే విలువలు ఇస్తే

x | 10 | 2 | 4 | 3 | 20 | అని సూచించవచ్చు.
0 1 2 3 4

దీని మొత్తాన్ని ఒక చరరాశిగా వ్యవహరించవచ్చు లేదా వేర్వేరు చరరాశులుగా కూడా తీసుకోవచ్చు. అంటే ఈ అయిదు విలువలలో దేనినైన మనం మార్చవచ్చు. అయితే ఒకసారి డిక్లేర్ చేసిన తర్వాత కొన్ని సందర్భాల్లో ఈ శ్రేణి పరిమాణాన్ని (విలువల సంఖ్యను) పెంచటం లేదా తగ్గించటం కుదరదు. ఇప్పుడు 10 మంది విద్యార్థుల మార్కులను తీసుకొని వాటి సగటును కనుగొనే విధానాన్ని చూద్దాం.

```
#include<stdio.h>
main()
{
    int mark[10], i, sum;
    float avg;
    for(i=0; i<10; i++)
        {
            printf("Enter marks for student no %d", i+1);
            scanf("%d", &mark[i]);
        }
    for(i=0; i<9; i++)
        {
            sum = sum+mark[i];
        }
    avg = (float) sum/10;
    printf("The average of 10 numbers is %f", avg),
}
```

ఈ ప్రోగ్రాం రన్ చేసి చూడండి ఈ కింది విధంగా వస్తుంది.

Enter the marks of student no 1

60

Enter the marks of student no 2

40

Enter the marks of student no 3

70

Enter the marks of student no 4

30

Enter the marks of student no 5

80

....

....

....

ఈ ప్రోగ్రాంలో కొన్ని ముఖ్యమైన అంశాలు, కొత్త అంశాలు ఉన్నాయి గమనించండి.

1. avg అనే చరరాశిని float గా డిక్లేర్ చేశాం. సగటు ఒక పూర్ణ సంఖ్యే వస్తుందని మనం ఖచ్చితంగా చెప్పలేం. దశాంశస్థానం కూడా రావచ్చు అందుకని float గా డిక్లేర్ చేశాం.

2. రెండు వలయాల్లోను నియమాలను i<10 అని మొదటి దాంట్లో, i<=9 అని రెండో దాంట్లో తీసుకున్నారు. నిజానికి ఈ రెండు నియమాలు ఒకటే i<10 అన్నా i<=9 అన్నా ఒకటే.

3. సగటుని గణించేటప్పుడు (float)sum/10 అని రాశాం. దీన్నే Type Casting అంటారు. avg అనే చరరాశి float లో ఉంది. కానీ sum అనే చరరాశి int లో ఉంది. కాబట్టి sum/10 ఫలితంకూడా int లోనే వస్తుంది. దీన్ని float లోకి మార్చటం కోసమే (float)sum/10 అని రాశాం.

4. avg అనే చరరాశిని ప్రింట్ చేయడానికి %f అని రాశాం. ఎందుకంటే avg అనే చరరాశిని float గా డిక్లేర్ చేశాం. float ని ప్రింట్ చేయాలంటే %f వాడాలని ముందు పాఠాల్లో చూశాం.

డేటాటైప్ ను మార్చటం : రెండు int చరరాశులతో భాగహారం చేస్తే అది మళ్లీ int రావాలన్న నిబంధన లేదు. ఉదాహరణకి 23/2 అన్న భాగహారంలో ఫలితం 11.5 అవుతుంది. కాని 23, 2 అనే విలువలు int చరరాశులయితే ఫలితం కూడా int అవుతుంది. అంటే ఫలితం 11 అనే వస్తుంది. ఒక float కి 23/2 ని అసైన్ చేసినా ఫలితం 11 వస్తుంది. ఎందుకంటే భాగహారం రెండు int ల మధ్యే జరుగుతోంది. దీని float గా మార్చాలంటే type castings ఉపయోగించాలి. (float) 11/5 అని రాస్తే ఫలితం float లో ఉంటుంది.

సంఖ్యలను క్రమపద్ధతిలో ఉంచటం (Sorting)

మనవద్ద ముగ్గురు విద్యార్థుల మార్కులు ఉన్నాయి. అవి 70, 60, 85. ఈ మూడు ఒక క్రమ పద్ధతిలో లేవు. వాటిని ఆరోహణ క్రమంలోకాని, అవరోహణ క్రమంలోకాని రాయాలి. అంటే 60 70 85 అనికాని 85 70 60 అనికాని

రాయాలి. ఇది చాలా తేలిక వాటి స్థానాలు మారిస్తే సరిపోతుంది. అయితే విలువలు మనకు తెలియదనుకోండి. అప్పుడు ఏంచెయ్యాలి? అంటే x, y, z అనే చరరాశులలో మూడు విలువలు తీసుకుంటాం. ఈమూడింటిని ఆరోహణ లేదా అవరోహణ క్రమంలో అమర్చాలంటే ఎలా? ఈ కింది ప్రోగ్రాం చూడండి. ఇందులో మూడు సంఖ్యలను తీసుకొని వీటిని ఆరోహణ క్రమంలో అమరుస్తున్నాం.

```
#include<stdio.h>
main()
{
    int x, y, z, temp;
    printf("Enter three numbers one by one");
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &z);
    if (x>y)
    {
/*inter change values of x and y*/
        temp = x;
        x = y;
        y = temp;
    }
    if(x>z)
    {
        /*inter change values of x and z */
        temp = x;
        x = z;
        z = temp;
    }
    if(y>z)
    {
        temp = y;
        y = z;
```

```
            z = temp;
        }
        printf("The numbers in ascending order are %d %d %d", x, y, z");
    }
```

పై ప్రోగ్రాంలో ముందుగా మూడు సంఖ్యలను ఇన్‌పుట్‌గా తీసుకుంటున్నాం. ఇవి ఆరోహణ క్రమంలో ఉండాలన్న నిబంధనేదీలేదు. ఏ క్రమంలోనైనా ఉండవచ్చు. వీటిని మనం ఆరోహణ క్రమంలో ఉంచేందుకు ప్రోగ్రాం రాయాలి. ఆరోహణ క్రమం అంటే ముందుగా చిన్న సంఖ్యలు తర్వాత పెద్ద సంఖ్యలుండాలి. ఇప్పుడు మొదటి సంఖ్యతో రెండవ సంఖ్యను పోల్చుతున్నాం. ఒకవేళ మొదటి సంఖ్య రెండవ దానికంటే పెద్దదైతే (x>y) ఆ చరరాశుల విలువలను పరస్పరం మారుస్తున్నాం. అప్పుడు మొదటి రెంటిలో చిన్నది ముందు పెద్దది తర్వాత ఉంటుంది.

తర్వాత మొదటి సంఖ్య, మూడో సంఖ్యకంటే పెద్దదైతే (x>z), ఆరెంటినీ పరస్పరం మార్చుతున్నాం. పై రెండు పోలికలు, మార్పులు వల్ల మనకు మొదటి చరరాశిలో మిగిలిన రెంటికన్నా చిన్న విలువ ఉంటుంది. ఇక రెండు, మూడు విలువలను పోల్చాలి. వాటిని కూడా పోల్చి, అవసరమైతే మార్పులు చేస్తున్నాం. దీనికారణంగా మనకు ముందు చిన్న విలువ, చివర అతి పెద్ద విలువ వచ్చాయి. అంటే ఆరోహణ క్రమంలో సంఖ్యలు వచ్చాయి.

మూడు సంఖ్యలలో మూడు పోలికలు మార్పులు వచ్చాయి. అదే 10 సంఖ్యలు, 100 సంఖ్యలు ఉంటే అన్ని పోలికలు ఉండాల్సిందేనా! అవును ఉండాలి కాని ప్రోగ్రాంను వలయాలు, శ్రేణులు ఉపయోగించి చేయటం వల్ల చాలా తక్కువ లైన్లలో సాధించవచ్చు. ఉదాహరణకి 10 సంఖ్యలను తీసుకుని వాటిని ఆరోహణ క్రమంలో రాసే ప్రోగ్రాంను చూద్దాం.

```
#include<stdio.h>
main()
{
    int num[10], i, temp, j;
    for(i=0; i<=9; i++)
    {
        printf("Enter a number \n");
        scanf("%d", &num[i]);
        /*number sorting in Ascending order*/
    }
    for(i=0; i<9; i++)
        for(j=i+1; j<=9; j++)
        {
                if(num[i]>num[j])
```

```
                {
                        temp = num[i];

                        num[i] = num[j];

                        num[j] = temp;

                }
        }
    printf("The numbers in ascending order are \n");
    for (i=0; i<=9; i++)
    {
        printf("%d \n", num[i]);
    }
}
```

ఈ ప్రోగ్రాం రన్ చేసి చూడండి. ముందు 10 సంఖ్యలను ఇన్‌పుట్ తీసుకుని వాటిని ఆరోహణ క్రమంలో ప్రింట్ చేస్తుంది. ఈ ప్రోగ్రాం లాజిక్‌ని అర్థం చేసుకోడానికి ప్రయత్నించండి.

పదాలను వాక్యాలను నిలవ ఉంచుకోవటం (Strings)

ఇప్పటి వరకూ మనం సంఖ్యలను నిలవ ఉంచుకునే ప్రక్రియలనే చూశాం. ఇప్పుడు పదాలను, వాక్యాలను నిలవ ఉంచుకునే ప్రోగ్రాంలను చూద్దాం. శ్రేణి విధానంలో సంఖ్యలను ఒకదాని తర్వాత ఒకటి నిలవ ఉంచుకుంటాం. అదేవిధంగా ఒక శ్రేణిలో అక్షరాలను ఒకదాని తర్వాత మరొకటి నిలవ ఉంచితే అది ఒక పదం అవుతుంది కదా. ఉదాహరణకి ఈ కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int i;
    char name[10];
    for(i=0; i<=4; i++)
    {
        printf("Enter a character \n");
        scanf("%c", &name[i]);
    }
    for(i=0; i<=4; i++)
```

```
    {
        printf("%c", name[i]);
    }
}
```

ఈ ప్రోగ్రాంను రన్‌చేస్తే ఈ కింది విధంగా వస్తుంది.

```
Enter a character
H
enter a character
E
Enter a character
L
Enter a character
L
Enter a character
O
HELLO
```

అయితే ఒక అక్షరం ఒకసారిచొప్పున ఎన్ని సార్లు ఎంటర్ బటన్ నొక్కాలి? విసుగ్గా ఉంది కదూ! దీనికోసమే 'సి' లో స్ట్రింగ్ (String) అనే విధానం ఉంది. దీన్ని ఉపయోగిస్తే పదం మొత్తం ఒకేసారి ఇన్‌పుట్‌గా తీసుకోవచ్చు. ఈ స్ట్రింగ్‌ను ఇన్‌పుట్, ఔట్‌పుట్ అవసరాలకు %s తో వ్యవహరిస్తారు.

ఉదాహరణకి ఈ కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    char name[10];
    printf("Enter your name");
    scanf("%s", name);
    printf("Your name is %s", name);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే ఈ విధంగా వస్తుంది.

Enter your name

VENU

Your name is VENU

ఇక్కడ కూడా name[10] అనే శ్రేణి చరరాశిలో అక్షరాలు నిలవ ఉంటాయి.

కంప్యూటర్ ఈ కింద చూపిన విధంగా శ్రేణిలో పదాలను నిలువ చేస్తుంది.

V E N U \0

అన్ని అక్షరాలు పూర్తయ్యాక చివర '\0' అనే గుర్తు ఉంది గమనించండి. దీన్ని Null Character అంటారు. ప్రతి పదానికి చివర ఈ గుర్తు ఉంటుంది. అంటే ఈ గుర్తు కనిపిస్తే పదం ముగిసినట్లు అన్నమాట. మనం %s వాడినపుడు ఈ పదం ఎక్కడ ముగిసిందో కంపైలర్ కనుక్కోడానికి ఈ గుర్తు ఉపయోగపడుతుంది.

పై ప్రోగ్రాంలో ఇంకో ముఖ్యమైన అంశం ఉంది. scanf స్టేట్ మెంట్‌లో చరరాశి పక్కన శ్రేణిగుర్తు ఇవ్వలేదు. శ్రేణి మొత్తానికి కలిపి అక్షరాలను ఒకేసారి తీసుకుంటూన్నాం కాబట్టి ఈ విధంగా ఇవ్వక్కరలేదు. అదేవిధంగా చరరాశి ముందు & గుర్తు ఇవ్వలేదు. ఇది స్ట్రింగ్ కాబట్టి %s వాడుతున్నప్పుడు '&' గుర్తు ఇచ్చినా, ఇవ్వకపోయినా తేడాలేదు. అందువల్లే ఇక్కడ ఆ గుర్తు ఇవ్వవలసిన అవసరం లేదు.

పదాలను తీసుకునేటప్పుడు %s ఒక పదాన్నే తీసుకుంటుంది. ఆపదంలో అక్షరాలసంఖ్య శ్రేణి పరిమాణం (డిక్లరేషన్ సమయంలో బ్రాకెట్లో ఇచ్చే సంఖ్య) కంటే ఎక్కువ ఉండకూడదు. ఎక్కడ ఖాళీ స్పేస్ వచ్చిందో ఆ పదాన్ని అక్కడి వరకే నిలవ ఉంచుతుంది.

ఉదాహరణకి పై ప్రోగ్రాంలో

Enter your name అని వచ్చినప్పుడు

VENU GOPAL అని ఇన్‌పుట్ ఇచ్చామనుకోండి

అప్పుడు కూడా VENU వరకే నిలవ ఉంచుకుంటుంది.

అదే VENUGOPAL అని మధ్యలో ఖాళీ లేకుండా ఇస్తే

అప్పుడు మొత్తం పేరును నిలవ ఉంచుకుంటుంది.

ఒక చిత్రమైన ప్రోగ్రాం చేద్దాం. ఒక పేరును ఇస్తే అందులో ఎన్ని అక్షరాలు ఉన్నాయో కనుక్కోవాలి.

ఒక పేరును తీసుకుంటే అది మన శ్రేణి పరిమాణం అంత పొడవు ఉండాల్సిన అవసరంలేదు. శ్రేణి పరిమాణ 20 అని తీసుకుని అందులో 20 లోపల ఎన్ని అక్షరాల పదాన్నయినా ఉంచవచ్చు. కాబట్టి మనం పదం పొడవున ప్రత్యేకంగా కనుక్కోవాల్సిన అవసరం ఉంది.

ప్రతిపదం చివర '\0' అనే గుర్తు ఉంటుందని పైన చూశాంకదా! ఇప్పుడు మొదటి నుంచి ఆ గుర్తు వరకు ఎన్ని అక్షరాలు ఉన్నాయో చూస్తే సరిపోతుంది. ఆ పదం పొడవు ఎంతో చెప్పవచ్చు. ఈకింది ప్రోగ్రాంను చూడండి.

```
#include<stdio.h>
main()
{
      int i, len;
      char word[20];
      printf("Enter a word \n");
      scanf("%s", word);
      while (word[i]!='\0')
      {
            i++;
      }
      len = i;
      printf("length of word is %d", len);
}
```

ఈ ప్రోగ్రాంను రన్ చేసి ఫలితాన్ని పరిశీలించండి.

ఇక్కడ మనం శ్రేణి మొదటి నుంచి ఆ పదంలో ఏ అక్షరం '\0' అవుతుందో గమనిస్తూ i విలువలు పెంచుతున్నాం. word [i] విలువ '\0' కి చేరగానే i విలువను పెంచటం ఆపేశాం. దాంతో ఇన్‌పుట్ చేసిన పదంలో ఎన్ని అక్షరాలన్నాయో i విలువ కూడా అంతే అవుతుంది. అందువల్ల మనకు ఆపదంలో ఎన్ని అక్షరాలున్నాయో తెలుస్తుంది.

ఒక పేరును ఇన్‌పుట్‌గా తీసుకుని దాన్ని వెనక్కి తిప్పి రాసే ప్రోగ్రాంను ఇప్పుడు చూద్దాం.

```
#include<stdio.h>
main ()
{
      int i, j;
      char name [20];
      printf("Enter name to stop press ch+z\n");
      while((ch=getchar())!=EOF)
      {
            name[i] = ch;
            i++;
      }
            sz = i;
            printf("\n Reverse is \n");
```

```
        for(i=sz; i>0; i--)
        printf ("%c", name[i]);
    }
```

ఈ ప్రోగ్రాం రన్ చేస్తే

```
Enter name
SAISHIVA
 REVERSE IS
 AVIHSIAS
```

అని ప్రింటవుతుంది.

ఒక పదంలో ఎన్ని VOWELS ఉన్నాయో లెక్కపెట్టే ప్రోగ్రాంను ఇప్పుడు చూద్దాం.

(ఇంగ్లీషులో A E I O U అక్షరాలను Vowels అంటారు)

```
#include<stdio.h>
{
        int i, count, len;
        char name [20];
        printf("Enter a name \n");
        scanf("%s", name);
        while (name[i]!='\0')
        {
                if(name[i] =='a'||name[i]=='e'||name[i]=='i'||name[i]=='O'||name[i]=='U')
                count++;
                i++;
        }
        print f("The number of vowels is %d", count);
}
```

ఈ ప్రోగ్రాం ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

ఒక పేరును రన్ టైమ్‌లో తీసుకుని దాన్ని Upper case నుంచి Lower case కు మార్చాలి.

```
#include<stdio.h>
main ()
```

```
{
        int i;
        char name [20];
        print f("Enter name in capitals \n");
        scan f("%s", name);
        while (name[i]! = '\0');
        {
                name [i] = name [i]+32;
        }
        printf ("the name in lower case is %s\n", name];
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

Enter name in capitals

SIRISHA

The name in lower case is sirisha

ఇప్పుడు మరో ప్రోగ్రాం చూద్దాం.

సాధారణంగా కొన్ని రహస్య సమాచారాలను ఒక కోడ్ భాషలోకి రాసి పంపుతారు.

మామూలుగా ABCD.... అని ఉన్న అక్షరాల స్థానంలో 4 లేదా అయిదు అక్షరాల తర్వాత ఉన్న అక్షరాన్ని ఉంచుతాం. అంటే A స్థానంలో E ని, B స్థానంలో F ని, C స్థానంలో G ని. ఆ విధంగా కోడ్‌ను తయారు చేయవచ్చు. ఇప్పుడు మనం ఎంటర్ చేసిన పదానికి కోడ్ పదాన్ని ఇచ్చే ప్రోగ్రాంను చూద్దాం.

```
#include<stdio.h>
main ()
{
        int i;
        char word [20];
        printf("Enter a word \n");
        scanf ("%s", word);
        printf(The word you entered is \n%s\n", word);
        while (word [i]! = '\0")
```

```
        word [i]+=5;
        printf ("The coded word is \n %s \n", word);
    }
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే

```
Enter a word
MUMBAI
The word you entered is
MUMBAI
The coded word is
QYQFEM
```

అని ప్రింటవుతుంది.

2-డైమెన్షనల్ శ్రేణులు - ఇప్పటి వరకూ మనం ఒకశ్రేణిలో కొన్ని విలువలను నిలువచేయటం, వీటికి సంబంధించిన ప్రోగ్రాంలను చూశాం. అయితే ఇక్కడ ఇంకో సమస్య ఉంది. రెండుతరగతులు ఇచ్చి వాటిలో విద్యార్థులకు వారికి ఉన్న అన్ని సబ్జెక్టుల్లోను వచ్చిన మార్కులను నిలవ చేయాల్సి వచ్చింది. అపుడేం చెయ్యాలి? రెండు తరగతులకు రెండు చరరాశులను తీసుకుంటాం. అయితే సబ్జెక్టులను ఏంచెయ్యాలి? దీనికి సమాధానంగా 'సి' మనకు 2-డైమెన్షనల్, మల్టీడైమెన్షనల్ శ్రేణులను అందిస్తోంది. ఉదాహరణకి ఒక తరగతిలో 4 సబ్జెక్టులలో అయిదుగురు విద్యార్థులకు వచ్చిన మార్కులను చూద్దాం.

		సబ్జెక్టు			
		1	2	3	4
	1	40	70	60	55
	2	69	56	91	84
విద్యార్థి	3	52	76	87	64
	4	91	82	73	64
	5	62	83	55	77

ఈ విధంగా విద్యార్థుల మార్కులను సూచించటం చాలా తేలిక. ఇందులో మనకు 5 అడ్డువరసలు, 4 నిలువు వరసలు ఉన్నాయి. ఇంతకు ముందు ఒక అడ్డువరసను తీసుకుని mark[4] అని శ్రేణి చరరాశిగా వ్యవహరించాం. ఇప్పుడు కూడా అదే విధంగా చేయవచ్చు. అయితే ఇందులో ఎన్ని అడ్డువరసలు, ఎన్ని నిలువువరసలు ఉన్నాయో మనం ముందే పేర్కొనాలి. పై పట్టికలో 5 అడ్డు వరసలు, 4 నిలువు వరసలు ఉన్నాయి. వీటన్నిటినీ ఒకే చరరాశిలో నిలవుంచడానికి వీలుగా ఆచరరాశికి mark[5][4] అని డిక్లేర్ చేయవచ్చు. అంటే చరరాశి పేరు ఇచ్చి ఎన్ని అడ్డు వరసలు ఉన్నాయో

ముందు చెప్పి ఎన్ని నిలువు వరసలున్నాయో తర్వాత చెప్తాం. దీన్ని డిక్లేర్ చేయటం కూడా తెలికే పై పట్టికలోని చరరాశిని

int mark[5][4] అని డిక్లేర్ చేస్తాం.

ఇదే విధంగా పది మంది విద్యార్థులున్నారు. ఒక్కొక్కరి పేరులో 20 అక్షరాలున్నాయి. అప్పుడు ఈ విధంగా డిక్లేర్ చేస్తాం.

char stu-name [10][20];

అంటే పది మంది విద్యార్థులు ఒక్కొక్కరి పేరులో 20 వరకూ అక్షరాలు ఉండవచ్చుని దీని అర్థం.

పది మంది విద్యార్థుల పేర్లను ఇన్‌పుట్ తీసుకుని వాటిని తిరిగి ప్రింట్ చేసే విధానాన్ని ఇప్పుడు పరిశీలిద్దాం.

```
#include<stdio.h>
main()
{
    int i, row, col;
    char stu-name[10][20];
    for(i=0; i<=9; i++)
    {
    printf("Enter a nae \n");
    scanf("%s", stu-name[i]);
    }
/*now display names*/
    for(i=0; i<=9; i++)
    printf("%s \n", stu-name[i]);
}
```

పై ప్రోగ్రాంలో రెండో వలయం కింద ఒకే స్టేట్‌మెంట్ ఉందికాబట్టి బ్రాకెట్ ఇవ్వాల్సిన అవసరం లేదు. ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి చూడండి.

ఈ 2 డైమెన్షనల్ శ్రేణి ఒక మాత్రిక (Matrix)ను పోలి ఉంటుంది. అయితే మెమొరీలో మాత్రం అడ్డు వరుసలు, నిలువు వరుసలు ఉండవు. మొత్తం ఒకే వరుసలో ఉంటుంది.

int num[2] [3];

అనే శ్రేణిలో 2 అడ్డు వరుసలు, 3 నిలువు వరుసలు ఉంటాయి. ఈ శ్రేణిలో సంఖ్యలను నిలువ చేస్తే ఈ విధంగా ఉంటాయి.

```
10  12  14
21   5  34
```

ఇప్పుడు రెండు అడ్డు వరుసలు, 2 నిలువు వరుసల (2×2) మాత్రికను తీసుకుందాం. అందులోకి విలువలను ఇచ్చి, ఏ స్థానంలో ఏ విలువ ఉందో చూద్దాం.

```
#include<stdio.h>
main()
{
    int i, j, matrix[2][2];
    for (i=0; i<=1; i++)
    {
        for (j=0; j<=1; j++)
        {
            printf("Enter %d, %d element \n", i+1, j+1);
            scanf("%d", &matrix[i][j]);
        }
    }
        printf("\n The elements of matrix are \n");
    {
        for(i=0;i<=1;i++)
        {
            for (j=0; j<=1; j++)
            {
                printf("%d",matrix[i][j]);
            }
            printf ("\n");
            }
}
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే ఈ కింది Output వస్తుంది.

```
Enter 1, 1 element
5
Enter 1, 2 element
7
```

Enter 2, 1 element

12

Enter 2, 2 element

15

The elements of matrix are

5 7

12 16

పై ప్రోగ్రాం output ను రెండు లైన్లలో రావటం కోసం ఒక వలయం బయట న్యూలైన్ చిహ్నాన్ని ఉపయోగించాం.

ఇప్పుడు ఒక మాత్రిక (Matrix) లోని అన్ని సంఖ్యలకు ఒక అంకెను కలిపే విధానాన్ని చూద్దాం.

```
#include<stdio.h>
main()
{
    int i, j, const, num [2] [2];
    for (i=0; i<=1; i++)
    {
        for(j=0; j<=1; j++)
        {
            printf("Enter %d, %d element \n", i+1, j+1);
            scanf("%d", & num[i] [j]);
        }
    }
    printf("Enter the constant you want to ad \n");
    scanf("%d", &const);
    printf("The result after adding the constant in \n");
    for (i=0; i<=1; i++)
    {
        for (j=0; j<=1; j++)
        {
```

```
            printf("%d", num[i] [j]+const);
        }
            printf("\n");
        }
    }
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

రెండు మాత్రికలను తీసుకుని వీటిని హెచ్చవేసే విధానాన్ని చూద్దాం.

```
#include<stdio.h>
main()
{
    int i, j, k, mat_1[2] [2], mat_2[2] [2], res [2] [2];
    /* res in to for storing the result */
    for (i=0; i<=1; i++)
    {
        for (j=0; j<=1; j++)
        {
            printf("Enter %d, %d element of matrix -1 \n", i+1, j+1);
            scanf("%d", & mat 1[i] [j]);
        }
    }
    for(i = 0; i<=1; i++)
    {
        for(j=0; j<=1; j++)
        {
            printf ("Enter %d, %d element of matrix-2 \n", i+1, j+1);
            scanf("%d", mat_2 [i] [j]);
        }
    }
```

```
/* Nor the multiplication process starts here */
for (i=0; i<=1; i++)
{
    for(j=0; j<=1; j++)
    {
        for(k=0; k<=1; k++)
            res [i][j] = res[i][j]+mat_1[i][k]* mat_2[k][j];
    }
    res[i] [j] = 0;
}
printf("The result is \n");
for(i=0; i<=1; i++)
{
    for(j=0; j<=1; j++)
    {
        printf("%d", res[i] [j]);
    }
}
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

ఇప్పుడు 3×3 మాత్రికలను తీసుకుని హెచ్చవేయండి.

ఇదే ప్రోగ్రాంలో చిన్న మార్పులు చేస్తే సరిపోతుంది.

ఒక సంఖ్య (వెయ్యిలోపల)ను తీసుకుని దాన్ని అక్షరాల రూపంలో ప్రింట్ చేసే ఈ ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
#include<conio.h>
main()
{
long num,num1 =1, k, x;
char ones [20][10] = {"ONE", "TWO", "THREE". "FOUR", "FIVE", "SIX", "SEVEN",
EIGHT",                                   "NINE", "TEN", "ELEVEN", "TWELVE".
'THIRTEEN", " FOUR                              TEEN", "FIFTEEN", "SIXTEEN",
'SEVENTEEN", "EIGHTEEN",                                     "NINETEEN"},
tens[8][10] = {"TWENTY", "THIRTY", "FORTY", "FIFTY", "SIXTY", "SEVENTY",
                         "EIGHTY", NINTY"};
while(num1>0)
{
    clrscr();
    printf("Enter a number\n");
    scanf("%d", &num1);
    num=num1;
    if(num>999)
{
    k=num\1000;
    num=num%1000;
    if(k>19)
    x=k/10;
    k=k%10;
    printf("%s ", tens[x-2]);
}
    if(k)
    printf("%s", ones[k-1]);
    printf("THOUSAND");
}
    if(num>99)
{
    k=num/100;
    num=n%100;
```

```
printf("%s HUNDRED ", ones [k-1]);
if(num)
printf ("AND ");
}
if(num>19)
{
k=num/10;
num=num%10;
printf('%s", tens[k-2]);
}
if(num)
printf("%s", ones [num-1]);
getch();
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

అభ్యాసం

1. ఒక శ్రేణి చరరాశిలో 10 సంఖ్యలలను ఇన్‌పుట్‌గా తీసుకుని వాటి మొత్తాన్ని కనుక్కోండి.
2. ఒక శ్రేణి చరరాశి (Array Variable) లోకి 10 సంఖ్యలను తీసుకుని వాటిలో అతి పెద్ద, అతిచిన్న సంఖ్యలను కనుక్కోండి.
3. ఒక శ్రేణి చరరాశిలోకి 20 సంఖ్యలను తీసుకుని అవరోహణ క్రమంలో అమర్చి ప్రింట్ చేయండి.
4. ఒక శ్రేణి చరరాశిలోకి 10 సంఖ్యలను తీసుకోండి. మరో సంఖ్యను మరో చరరాశిలోకి ఇన్‌పుట్‌గా తీసుకోండి. రెండవసారి తీసుకున్న సంఖ్య ఈ శ్రేణిలో ఉందా? లేదా? ఉంటే ఎన్ని సార్లు ఉంది. అని కనుక్కునేందుకు ప్రోగ్రాం రాయండి.
5. ఒక పదాన్ని తీసుకుని దాన్ని తిరగదిప్పి ప్రింట్ చేసేందుకు ప్రోగ్రాం రాయండి.
6. ఒక పదాన్ని ముందుగా ఒక శ్రేణి చరరాశిలోకి తీసుకోండి. తర్వాత ఒక అక్షరాన్ని మరో చరాశిలోకి తీసుకోండి. ఇప్పుడు ఆ అక్షరం ముందు తీసుకున్న పదంలో ఎన్నిసార్లు వచ్చిందో తెలిపే ప్రోగ్రాం రాయండి.
7. ఒక 2-డైమెన్షనల్ శ్రేణి చరరాశిలోకి కొన్ని పేర్లను తీసుకుని, వాటి మొదటి అక్షరం ఆధారంగా ఆరోహణ క్రమంలో అమర్చండి (Sort).-

◆◆◆

అధ్యాయం 9

ప్రమేయాలు (Functions)

ఒక అంశం గురించి వంద పేజీల నోట్సు రాయాల్సిన పని మీమీద పడింది. అది కూడా రెండు రోజుల్లోనే రాయాలి. అప్పుడు ఏం చేస్తారు? కష్టంగా ఉన్న అంశాలను, శ్రద్ధ పెట్టాల్సిన అంశాలను మీరేరాసి, తేలిగ్గా ఉన్నవి ప్రధాన అంశానికి సంబంధం ఎక్కువగా లేని అంశాలను మీ తమ్ముడు, చెల్లెలు లేదా ఫ్రెండ్‌కో ఇచ్చి రాయమంటారు. ఆ విధంగా మీపై పని భారం తగ్గటమేకాక పని తొందరగా పూర్తవుతుంది.

ఇదే విధంగా ప్రోగ్రామింగ్‌లో కూడా మన ప్రధాన ప్రోగ్రాంలోనే అన్ని అంశాలను రాసే అవకాశం మనకి ఉన్నప్పటికి అందులో కొన్ని ఇబ్బందులున్నాయి.

ఉదాహరణకి ఒక ప్రోగ్రాంలో నాలుగు లైన్లను ప్రింట్ చేయాలి. ఆ నాలుగు లైన్లను కనీసం పదిసార్లు ప్రింట్ చేయాలి. అయితే ప్రతి ప్రింట్‌కి మధ్య ప్రోగ్రాంకు సంబంధించిన వేరే స్టేట్‌మెంట్‌లు ఉంటాయి. ఉదాహరణకు స్టూడెంట్స్ మార్కులే కాకుండా, వాటికి సంబంధం లేని వారే స్టేట్‌మెంట్‌లు కూడా ప్రోగ్రాంలో రాయాల్సిన అవసరం వుంటుంది. అంటే వలయం ఉపయోగించే వీలులేదు. అందువల్ల ఈ నాలుగు లైన్లను పదిసార్లు - అంటే 40 లైన్లు రాయాల్సిందేనా? అవసరంలేదు. ఈ కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int i, num;
    char name[30];
    printf("Enter student number \n")
    scanf(%d", &num);
    printf("Write the poem \n");
    printf("Rain rain go away \n");
    printf("Little Johny wants to play \n");
    printf("This poem is good \n");
    printf("Enter another student number \n");
```

```
scanf("%d", &num);
printf("Enter name \n");
scanf("%s", name);
printf("Enter age");
scanf("%d", &i);
printf("Tell poem");
printf("Rain rain go away \n");
printf("Little Johny wants to play \n");
printf("Tell the poem again \n");
printf("Rain rain go away \n");
printf("Little Johny wants to play \n");
printf("OK good \n");
}
```

పై ప్రోగ్రాంలో poem చెప్పిన రెండు లైన్లను మూడు సార్లు రాశాం. వాటిని వలయంలో పెట్టే అవకాశం కూడా లేదు. అందువల్ల వీటిని ఒక చోట రాసి వేర్వేరు చోట్ల అవే స్టేట్మెంట్లను పిలిచే వీలుంటే బాగుంటుంది కదా! అంటే ఇన్నిసార్లు రాయాల్సిన అవసరం లేకుండా ఎక్కడికి కావాలంటే అక్కడికి ఈ లైన్లను పిలవగలిగితే బాగుంటుంది. ప్రమేయాల (functions) ను ఉపయోగించి దాన్ని సాధించవచ్చు. 'సి' భాష ఈ ప్రమేయాలకు అవసరమైన వాతావరణాన్ని కల్పిస్తోంది. ఏ స్టేట్మెంట్లనైతే మనం పిలవదలచుకున్నామో వాటిని ఒక పక్కన రాసి ఒక ప్రధాన ప్రోగ్రాంలో వాటిని పిలవచ్చు. ఉదాహరణకి

```
#include<stdio.h>
main()
{
    int i, num;
    char name[20];
    void call_poem();  ← డిక్లేర్ చేయటం
    printf("Enter student number \n");
    scanf(%d", &num);
    printf("Enter name \n");
    scanf("%s", name);
    call_poem();            ← ప్రమేయాన్ని పిలవటం
    printf("OK");
```

```
        printf(Enter name \n");
        scanf("%s", &name);
        call_poem();
        printf("Tell again");
        call_poem();            ←  ప్రమేయాన్ని పిలవటం
        printf("OK \n");
    }
    void call_poem()            ←  ప్రమేయాన్ని నిర్వచించటం
    {
        printf("Rain rain go away \n");
        printf("Little Johny Wants to Play \n");
    }
```

ఈ ప్రోగ్రాంను పరిశీలించండి. ఇందులో poem ను main() బయట రాశాం. ప్రోగ్రాం చివర మాత్రమే రాశాం. అయితే దాన్ని కూడా ప్రమేయంలో ఉంచాం. ఇందులో call_poem అనే చరరాశిని డిక్లేర్ చేసేటప్పుడు void అని ఉపయోగించాం. అంటే అది ఏ డేటాటైప్‌కు చెందనిదని అర్థం. ఇక చరరాశి చివర () బ్రాకెట్లు పెట్టాం. ప్రమేయానికి గుర్తు ఇది. మామూలు చరరాశికి, ప్రమేయానికి తేడా ఈ బ్రాకెట్లు.

poemను ప్రింట్ చేయాల్సినప్పుడల్లా call_poem(); అని రాశాం. అంటే ఆ ప్రమేయాన్ని అక్కడికి పిలుస్తున్నామన్నమాట. అప్పుడు ప్రమేయంలో ఉన్న కోడ్ ఈ call_poem(); అని ఉన్న చోట ఎగ్జిక్యూట్ అవుతుంది. అంటే ఈ ప్రమేయంలో ఉన్న రెండు స్టేట్‌మెంట్లూ call_poem(); అని ఉన్న చోట ప్రింటవుతాయి. అంటే ఆ రెండు స్టేట్‌మెంట్‌లకు బదులుగా call_poem(); అనిరాస్తే సరిపోతుంది. ఇదేవిధంగా ఎన్ని లైన్లకోడ్ ఉన్నాకూడా ఒకేస్టేట్‌మెంట్‌తో ప్రమేయాన్ని పిలిస్తే సరిపోతుంది. ఇప్పుడు ప్రమేయాల ప్రాధాన్యం మీకు అర్థమై ఉంటుంది. ప్రోగ్రాంలో 200 లైన్లున్న విభాగం 10 సార్లు తిరిగి వచ్చిందనుకుందాం. మొత్తం 2000 లైన్లయింది. అదే ఆ 200 లైన్ల కోడ్‌ను ప్రమేయంలో రాస్తే 10 చోట్ల కేవలం 10 లైన్లను ఉపయోగించి ప్రమేయాన్ని పిలిస్తే సరిపోతుంది. అంటే ప్రమేయంలో 200 లైన్లు, ప్రోగ్రాంలో 10 లైన్లు మొత్తం 210 లైన్ల ప్రోగ్రాం రాస్తే సరిపోతుంది.

ప్రమేయాన్ని ఎలా ఉపయోగించాలో చూద్దాం. పై ప్రోగ్రాంలో చూసినట్లుగా ముందు ప్రమేయాన్ని డిక్లేర్ చెయ్యాలి. అయితే ప్రస్తుతానికి void ను ఉపయోగించి డిక్లేర్ చేయవచ్చు. పేరు చివర బ్రాకెట్లు () తప్పకుండా ఉండాలి. కోడ్ అవసరమైన చోటల్లా ఇదే ప్రమేయం పేరును రాయాలి. అయితే డిక్లరేషన్‌లో ఉన్నట్లుగా void అని ఉండదు. ప్రమేయాన్ని ఉపయోగించి ఒక ప్రోగ్రాంను ఇప్పుడు చూద్దాం.

```
#include<stdio.h>
main()
{
    int num, x, y;
    void print_line();
```

```
    printf("The first instance is stated here");
    print_line();
    printf("Second instance is quoted here");
    print_line();
}
void print_line()
{
    printf("Quick Brown Fox Jumps Over Lazy Dog");
}
```

main() అయిపోయిన తర్వాత ఈ ప్రమేయాన్ని నిర్వచించి ఆ కోడ్‌ను అందులో రాస్తాం. ఇదికూడా main() లాగానే విడిగా ఉంటుంది. ఇప్పుడు మరోసారి ప్రోగ్రాంను పరిశీలించండి. main() కూడా ఒక ప్రమేయమే. అయితే ఈ ప్రమేయాన్ని మనం ఎక్కడా నిర్వచించాల్సిన అవసరం లేదు. ఎందుకంటే 'సి' ప్రోగ్రాం main() తోనే ప్రారంభమవుతుంది. ఇతర ప్రమేయాలన్నిటినీ మనం main() లో తప్పని సరిగా డిక్లేర్ చెయ్యాలి.

కొన్ని ప్రమేయాలను మనం ఇప్పటికే ఉపయోగిస్తున్నాం. ఉదాహరణకు printf, scanf వంటి స్టేట్‌మెంట్‌లు మనం ఉపయోగిస్తున్న ప్రమేయాలే. అయితే అవి సిస్టమ్‌లోని లైబ్రరీలో ఉంటాయి. వాటిని లైబ్రరి ప్రమేయాలు లేదా లైబ్రరి ఫంక్షన్స్ (Library Functions) అంటారు. వీటిని కూడా ఎక్కడో డిక్లేర్ చేసి, నిర్వచించి ఉండాలి. అప్పుడే వాటిని మనం ఇక్కడ పిలవగలుగుతున్నాం. మరి వీటిని ఈ ప్రోగ్రాంలో ఎక్కడా డిక్లేర్ చేయలేదు, నిర్వచించలేదు మరి ఏవిధంగా ఉపయోగిస్తున్నాం? ఈ అనుమానం మీకు ఈ పాటికి వచ్చే ఉంటుంది. ప్రతి ప్రోగ్రాం పైభాగంలోను #include<stdio.h> అని రాస్తున్నాం చూశారా. stdio.h అనేది ఒక ఫైలు అందులోను ఒక 'సి' ప్రోగ్రాం ఉంటుంది. దానిలోనే మనం ఈ ప్రమేయాలను నిర్వచిస్తున్నాం. కాబట్టే ఈ రెండు ప్రమేయాలను ఉపయోగించే ప్రతిచోట మనం తప్పని సరిగా #include<stdio.h> రాయాల్సిందే.

ప్రమేయాల్లో రెండు రకాలున్నాయి. మొదటివి లైబ్రరీ ప్రమేయాలు కాగా రెండవ రకం User నిర్వచించిన ప్రమేయాలు(User defined functions).

ఇక్కడ User అంటే ప్రోగ్రాం రాసేవారు, అంటే మనమే. పైన మనం నిర్వచించిన ప్రమేయాలనే User defined functions అంటారు.

కొన్ని లైబ్రరీ ప్రమేయాలను కూడ పైన చూశాం. మరికొన్నిటిని ఇప్పుడు చూద్దాం. ఒక పదంలో ఎన్ని అక్షరాలున్నాయో కనుక్కోవటానికి ఇంతకుముందు ఒక ప్రోగ్రాం రాశాం. కాని దీనికి సంబంధించిన ఒక లైబ్రరీ ప్రమేయాన్ని ఉపయోగించి పదం పొడవు (String length) ను తెలుసుకోవచ్చు. ఇందుకోసం strlen() అనే ప్రమేయాన్ని ఉపయోగించాలి. ఈ ప్రమేయం string.h అనే ఫైల్‌లో ఉంటుంది కాబట్టి దాన్ని కూడా మన ప్రోగ్రాంలో ఉపయోగించాలి. కింది ప్రోగ్రాంలో మనం ఈ ప్రమేయం ఉపయోగం తెలుసుకుందాం.

```
#include<stdio.h>
#include<string.h>
main()
```

```
{
    int count;
    char name[20];
    printf("Enter name \n");
    scanf("%s", name);
    count i = strlen(name);
    printf("You have %d letters in your name", count);
}
```

పై ప్రోగ్రాం రన్ చేస్తే

```
Enter name
Saisivapublications
You have 19 letters in your name
```

ఈ ప్రోగ్రాంలో strlen() అనే ప్రమేయం లోపల name అనే చరరాశిని ఇచ్చాం. string.h అనే ఫైల్‌లో ఈ ప్రమేయాన్ని నిర్వచించాం. ఇక్కడ ఇచ్చిన name అనే చరరాశి ఆ ప్రమేయం లోకి వెళ్లి అక్కడి నుంచి string పొడవు తిరిగి వస్తుంది.

ఒక చరరాశిలోని పదాన్ని మరొక చరరాశిలోకి copy చేయవచ్చు. సాధారణంగా రెండు int చరరాశులు ఉంటే, వాటి మధ్య = గుర్తుతో విలువలను ఒకదాని నుంచి మరొక చరరాశిలోకి copy చేయవచ్చు. కాని string విషయంలో అలాకుదరదు. అందువల్ల strcpy() ప్రమేయాన్ని ఉపయోగించి దీన్ని సాధించవచ్చు. నిజానికి రెండు strings అంటే రెండు శ్రేణులేకదా. ఒక శ్రేణిలోని ప్రతి అక్షరాన్ని మరో శ్రేణిలోకి copy చెయ్యాలి. ఇందుకు బదులుగా strcpy ప్రమేయాన్ని ఉపయోగిస్తే సరిపోతుంది.

ఈ strcpy ప్రమేయం కూడా string.h ఫైల్‌లోనే ఉంటుంది. కింది ఉదామరణలో ఈ ప్రమేయాన్ని వివరంగా చూద్దాం.

```
#include<stdio.h>
#include<string.h>
main()
{
    char name_1[20], name_2[20];
    printf("Enter name \n");
    scanf("%s", name_1);
    strcpy(name_2, name_1);
    printf("The original name is %s \n", name_1);
    printf("The copied name is %s \n", name_2);
}
```

ఇప్పటివరకూ string కి సంబంధించిన ప్రమేయాలను చూశాం. ఇన్‌పుట్, ఔట్‌పుట్‌లకు సంబంధించిన మరికొన్ని ప్రమేయాలను ఇప్పుడు చూద్దాం. ముందుగా తెరపై ఉన్న సమాచారాన్నంతా క్లియర్‌చేసే clrscr() ప్రమేయాన్ని చూద్దాం. ఈ ప్రమేయాన్ని ఉపయోగించాలంటే conio.h అనే హెడర్‌ఫైల్‌ను ఉపయోగించాలి.

```
#include<stdio.h>
#include<conio.h>
main()
{
    clrscr();
    printf("Now you can see the fresh screen");
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే ముందు మీ కంప్యూటర్ తెర మొత్తం ఖాళీఅవుతుంది. తర్వాత

Now you can see the fresh screen

అని ప్రింటవుతుంది.

సాధారణంగా ప్రోగ్రాంను రన్ చేసేటప్పుడు తెరక్లియర్‌గా ఉండాలనుకుంటే ఈ ప్రమేయాన్ని ఉపయోగిస్తారు. ఈ clrscr() ప్రమేయం కూడా డిక్లరేషన్‌ల తర్వాతే ఉండాలి.

ఇప్పుడు మరికొన్ని ఇన్‌పుట్ ఔట్‌పుట్ ప్రమేయాలను పరిశీలిద్దాం.

(i) getch(): ఒక ఒక అక్షరాన్ని ఇన్‌పుట్‌గా తీసుకోవాలన్నప్పుడు ఈ ప్రమేయాన్ని ఉపయోగిస్తాం. ఈ ప్రమేయాన్ని ఉపయోగించినప్పుడు ఇన్‌పుట్ తర్వాత

Enter any character

అని వచ్చినచోట k ఇవ్వాలనుకుందాం. మామూలుగా k ఉన్న కీని నొక్కి ఆ తర్వాత Enter కీని నొక్కుతాం. అయితే, getch() ప్రమేయం ఉపయోగించినప్పుడు ఈ Enter కీని నొక్కాల్సిన అవసరం ఉండదు. కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
#include<conio.h>
main()
{
    char ch;
    clrscr();
    printf("enter a characer \n");
    ch = getch();
    printf("The character you entered is %c \n", ch);
}
```

ఈ ప్రోగ్రాంను రన్ చేస్తే

Enter a character

k   కేవలం k ఉన్న కీనినొక్కితే చాలు Enter అవసరంలేదు.

The character you etered is k

పై ప్రోగ్రాంలో getch() ని ఉపయోగించక తీరుని పరిశీలించండి. getch() వల్ల మరో ఉపయోగం కూడా ఉంది సాధారణంగా ప్రోగ్రాం ఎగ్జిక్యూషన్‌ను మధ్యలో ఆపి తిరిగి రన్ చేయాలనుకుంటే ఈ getch() ని ఉపయోగిస్తాం. ఉదామరణకి కింది ప్రోగ్రాంను పరిశీలించండి.

```
#include<stdio.h>
#include<conio.h>
main()
{
    int i;
    for(i=1; i<=10; i++)
    {
        printf("%d \n", i);
        printf("Press any key to continue \n");
        getch();
    }
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే కింది విధంగా వస్తుంది.

```
1
Press any key to continue
2
Press any key to continue
3
Press any key to continue
4
Press any key to continue
5
```

Press any key to continue

6

Press any key to continue

7

Press any key to continue

8

Press any key to continue

9

Press any key to continue

10

Press any key to continue

పై ప్రోగ్రాంలో ఒక అంకె ప్రింట్ అయ్యాక కింద

Press any key to continue

అని ప్రింటవుతుంది. అంటే getch() వచ్చినప్పుడు ప్రోగ్రాం ఆగుతుంది. ఏదైనా కీని నొక్కగానే తిరిగి ప్రోగ్రాం రన్ అవుతుంది.

getchar()- ఇది కూడా getch() వంటి ప్రమేయమే అయితే getchar() ను ఉపయోగించినప్పుడు ఒక కీని నొక్కి ఆ తర్వాత Enter కీని కూడా నొక్కాలి. అప్పుడే ఇన్‌పుట్ జరుగుతుంది. ఈ ప్రమేయం కూడా కేవలం ఒక అక్షరాన్ని మాత్రమే తీసుకుంటుంది.

gets()- ఇంతకుముందు ఒక పేరుని లేదా పదాన్ని (string ని) ఇన్‌పుట్‌గా తీసుకోవాలంటే scanf ప్రమేయంలో %s ఉపయోగించాం. అయితే ఖాళీ (space) వస్తే %s ఆగిపోతుంది. అంటే స్పేస్‌కు ముందున్న అక్షరాలనే అది తీసుకుంటుంది. అదే gets() ప్రమేయాన్ని ఉపయోగిస్తే space వచ్చినా, మరే ఇతర గుర్తు వచ్చినా, Enter నొక్కేవరకూ తీసుకుంటుంది.

c is a powerful language అనే వాక్యాన్ని %s ఉపయోగించటం ద్వారా తీసుకోలేము అదే gets() ప్రమేయంతో ఇన్‌పుట్‌గా గ్రహించవచ్చు. కింది ప్రోగ్రాంతో gets() ఉపయోగాన్ని చూద్దాం.

```
#include<stdio.h>
#include<conio.h>
main()
{
    char word[20];
```

```
        printf("Enter a setence \n");
        gets(word);
        printf("You entered \n %s \n", woid)
    }
```

ఈ ప్రోగ్రాం రన్ చేస్తే

Enter a sentence

practice makes us perfect

You entered practice makes us perfect

ఈ విధంగా మనం వాక్యాలను కూడా ఇన్‌పుట్‌గా తీసుకోవచ్చు. ఇవికాక అనేక లైబ్రరీ ప్రమేయాలున్నాయి. TURBO C లోని Helpen వీటిని చూడవచ్చు.

ప్రమేయం నుంచి విలువలను పొందటం: ఇప్పటివరకూ ప్రమేయాలు స్వతంత్రంగా ఎగ్జిక్యూట్ కావటం చూశాం. ప్రధాన ప్రోగ్రాం నుంచి వాటిని పిలిచే సదుపాయమేగాని మరీ ఇతర సంబంధం లేదు. ప్రమేయంలో కొన్ని గణనలు పూర్తి చేసి వాటి ఫలితాన్ని ప్రధాన ప్రోగ్రాంకు అందించే సదుపాయాన్ని 'సి' మనకు కల్పిస్తోంది. ఉదామరణకు 10 సంఖ్యల మొత్తాన్ని, సగటును మనం ప్రమేయంలో గణించి ఆన్ని ప్రధాన ప్రోగ్రాంలో ప్రింట్‌చేసే ప్రక్రియ ఇప్పుడు చూద్దాం.

```
#include<stdio.h>
main()
{
    float avg;
    float call_value();
    avg = call_value();
    printf("The average of first 10 number is %d", avg);
}
    float call_value()
{
    int i, sum, average;
    for(i=1; i<=10; i++)
    {
        sum = sum+i;
    }
    average = (float) sum/10;
```

```
    return(average);
    }
```

పై ప్రోగ్రాంలో కొన్ని అంశాలను పరిశీలిద్దాం.

1. ప్రమేయాన్ని డిక్లేర్ చేసే సమయంలో void కి బదులుగా float ని ఉపయోగించాం. ఈ ప్రోగ్రాంలో మనం ఒక float విలువను ప్రమేయంనుంచి main() కు return చేస్తున్నాం. అందువల్ల మనం float ని ఉపయోగించి డిక్లేర్ చేశాం.
2. ప్రోగ్రాంలో ప్రమేయాన్ని పిలిచేటప్పుడు avg=call-valued() అని ఇచ్చాం. అంటే ప్రమేయంలోంచి return ద్వార వెనక్కి వచ్చే విలువ avg లోకి చేరుతుంది. అంటే call-value() అని ఉన్నచోటికి return ద్వార వచ్చే విలువ చేరుతుంది.
3. ప్రమేయంలో మూడు చరరాశులను డిక్లేర్ చేశాం. ఆ ప్రమేయంలో కూడా main() లో మాదిరిగానే విలువలను డిక్లేర్ చేయవచ్చు. అయితే ఈ చరరాశుల పరిమితి ప్రమేయాన్ని దాటి వెళ్లదు. అంటే ప్రమేయంలో డిక్లేర్ చేసిన చరరాశులను main() లో ఉపయోగించలేము. అదేవిధంగా main() లో డిక్లేర్ చేసిన వాటిని ప్రమేయంలో ఉపయోగించలేము.
4. ప్రమేయం చివర్లో return అని ఉపయోగించాం. అంటే ఇక్కడ గణించిన విలువను ప్రధాన ప్రోగ్రాంకు పంపుతున్నాం. అందుకు return ని ఉపయోగిస్తాం. ఈ return ద్వార పంపే విలువను తీసుకోవడానికి మనకు ప్రధాన ప్రోగ్రాంలో వీలు ఉండాలి. పై ప్రోగ్రాంలో మనం avg అనే చరరాశిలోకి ఈ విలువను పంపుతున్నాం.

మరో ప్రోగ్రాంను చూద్దాం.

```
#include<stdio.h>
main()
{
    int i, sum;
    int call_sum();
    sum = call_sum();
    printf("The sum of first 50 numbers is %d", sum);
}
int call_sum()
{
    int i, sum;
    for(i=1; i<50; i++)
    {
        sum = sum+i;
```

```
        return (sum);
    }
```

ఈ ప్రోగ్రాంను రన్ చేసి ఫలితం చూడండి.

ప్రమేయానికి విలువలను పంపటం

ఒక ప్రమేయం నుంచి ప్రధాన ప్రోగ్రాంకు విలువను ఎలా పంపాలో చూశాం. ఇప్పుడు ప్రధాన ప్రోగ్రాం నుంచి ప్రమేయానికి విలువను ఎలా పంపాలో చూద్దాం. ఈ కింది ప్రోగ్రాంకు పరిశీలించండి.

```
#include<stdio.h>
main()
{
    int i, avg, end_num;
    int call avg(int a);
    printf("Enter a number up to which you want average"),
    scanf("%d", &end_num);
    avg = call-avg(end_num);
    printf("The average is %f", avg);
}
    int call_avg(int a)
{
    int count = 0;
    int sum, float avg;
    for(i=1; i<=a, i++)
    {
        sum = sum+i, count++;
        avg = (float) sum/count;
    }
    return(avg);
}
```

ఈ ప్రోగ్రాంలో కొన్ని అంశాలను పరిశీలిద్దాం.

1. ప్రమేయాన్ని డిక్లేర్ చేసే సమయంలో ప్రమేయం చివరి బ్రాకెట్‌లో int a అనే చరరాశిని డిక్లేర్ చేశాం. అంటే ఆ ప్రమేయానికి a అనే చరరాశిలో విలువను పంపుతున్నామని అర్థం. ఇప్పుడు కింద ప్రమేయాన్ని నిర్వచించే సమయంలో ఇదే డిక్లరేషన్ చేస్తున్నాం. అంటే పైన విలువను ఆ చరరాశిలోకి మనం విలువను తీసుకుంటున్నాం.
2. ప్రమేయాన్ని పిలిచే సమయంలో బ్రాకెట్‌లు ఒక చరరాశికి ఉంచాం. అంటే ఇక్కడి విలువను కింది ప్రమేయంలోకి పంపుతున్నామని అర్థం. ఇక్కడ చరరాశిలో విలువ (end_num లో విలువ) ప్రమేయంలోని చరరాశిలోకి copy

3. (i) ఇప్పుడు కింద a అనే చరరాశిలో ఏవైనా మార్పులు జరిగినా, అవి పైన ఉన్న end_num చరరాశిలో ప్రతిఫలించవు. అంటే a విలువ మారినా end_num విలువ మారదు.

ప్రమేయాలకు సంబంధించి మరికొన్ని ప్రోగ్రాంలను ఇప్పుడు చూద్దాం. ముందుగా ఏ సంవత్సరంలో పుట్టారో ఇన్‌పుట్‌గా ఇస్తే ఆ వ్యక్తి వయస్సును ప్రింట్ చేసే ప్రోగ్రాం చూద్దాం.

```
#include<stdio.h>
main ()
{
    int dob, i;
    char name [20];
    void get_age (int a);
    printf("Enter name \n");
    scanf("%s", name)
    printf("Enter year of birth \n");
    scanf("%d", &dob);
    printf("Hello %s, your age in years is", name);
    get_age (dob);
}
void get_age (int a)
{
    printf ("%d", 2000_a);
}
```

ఈ ప్రోగ్రాం ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి. ఒక సంఖ్యకు factorial కనుగొనడానికి ప్రమేయాలను ఉపయోగించి ఈ కింది విధంగా ప్రోగ్రాం రాయవచ్చు.

```
#include<stdio.h>
main()
{
    int n;
    long int f;
    long int fact(int n);
    printf("\n enter a number\n");
    scanf("%d",&n);
    f=fact(n);
    printf("\n factorial of %ld\r.",f);
```

```
        getch();
    }
        long int fact(int n)
    {
        int f=1,i;
        for(i=n;i>=1;i—)
        {
            f=f*i;
        }
        return(f);
    }
```

ఈ ప్రోగ్రాం ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి. ఈ ప్రోగ్రాంలో వలయాన్ని ఉపయోగించకుండా ప్రమేయం లోపల ప్రమేయం రాయటం ద్వారా factorial ను కనుగొన్నాం. ఈ పద్ధతినే Recursion అంటారు.

అభ్యాసము

1. ఒక వృత్తం వ్యాసార్థాన్ని (రేడియస్‌ను) ప్రధాన ప్రమేయంలో user నుంచి తీసుకుని, ఆవృత్త వైశాల్యం కనుగొనటానికి ఒక ప్రమేయాన్ని రాయండి. ఈ ప్రోగ్రాంతో వ్యాసార్థాన్ని main()లో తీసుకుని వైశాల్యాన్ని area() అనే ప్రమేయంలో రాయాలి.

2. ఒక తరగతిలో విద్యార్థులకు సబంధించి తెలుగు, లెక్కలు, సైన్స్‌లతో మార్కులను తీసుకునేందుకు ఒక ప్రమేయాన్ని రాయండి. ఈ ప్రమేయాన్ని main() లో పిలిచినప్పుడల్లా ఆవివరాలను తీసుకోవాలి. అదే ప్రమేయంలో ప్రతి విద్యార్థికి మొత్తం మార్కులు, మార్కుల శాతం కూడా గణించి చెప్పాలి.

3. ఒక ఉద్యోగి జీతాన్ని BASIC, TA, DA, HRA ఇన్‌పుట్ తీసుకుని వాటిని ఒక ప్రమేయంలోకి Arguments గా పంపాలి. అప్రమేయంలో ఉద్యోగి TAX ను, Salary ని కనుగొనాలి. Tax కనుక్కోడానికి ఈ కింది ఫార్ములాను ఉపయోగించాలి.

if basic > 2000

Tax = 20% of basic

if basic > 3000

Tax = 30% of basic

if basic > 5000

Tax = 45% of basic

✦✦✦

అధ్యాయం 10

పాయింటర్లు (Pointers)

చరరాశులలోని విలువలను మెమోరీలో నిలువచేస్తామని ముందే తెలుసుకున్నాం. ప్రతి విలువకూ మెమోరీలో ఒక్కో స్థానం ఉంటుంది. కంప్యూటర్ మెమోరీలో ఒక్కో స్థానానికి ఒక్కో చిరునామా (Address) ఉంటుంది. మనకి ప్రతి ఇంటికీ ఒక్కోడోర్ నంబర్ ఉన్నట్లుగానే కంప్యూటర్‌లో ఒక్కో స్థానానికి ఒక్కో చిరునామా కేటాయించబడుతుంది. ఈ చిరునామా సంఖ్యల రూపంలో ఉంటుంది. అంటే ఒక్కో చిరునామాను ఒక్కో సంఖ్యతో సూచిస్తారు. ఉదాహరణకి

```
int a;
```

అది ఒక చరరాశికీ డిక్లేర్ చేసినప్పుడు ఆ చరరాశికి మెమోరీలో ఒక స్థానం, దానికి ఒక చిరునామా కేటాయించబడుతుంది. నిజానికి ఇప్పటివరకూ మనం చరరాశులను వాటి పేర్లతోనే వ్యవహరించాం. చరరాశుల లేదా మెమోరీలోని విలువలతో చేసే ప్రక్రియలన్నిటికీ పేర్లనే ఉపయోగించాం. అయితే ఇందులో కొన్ని ఇబ్బందులున్నాయి. ముఖ్యంగా ప్రమేయాలను ఉపయోగించేటప్పుడు ఈ ఇబ్బందులు బయటపడతాయి. ఉదాహరణకి ఈ కింది ప్రోగ్రాం చూడండి.

```
#include<stdio.h>
main()
{
    int a, b:
    void swap (int a, int b);
    a=10;
    b=20;
    swap (a,b);
    printf("The values after swapping are %d, %d ", a,b);
}
void swap(int a, int b)
{
    int temp;
```

```
        temp = a;
        a = b;
        b = temp;
    }
```

ఈ ప్రోగ్రాంలో మనం ఏంచేశామో చూడండి.

ముందుగా ప్రధాన ప్రోగ్రాంలో a, b అనే చరరాశులను డిక్లేర్ చేసి వాటికి 10, 20 విలువలను ఇచ్చాం. ఇప్పుడు ప్రమేయంలోకి ఈ a, b విలువలను పంపాము. ప్రమేయంలోపల విలువలను పరస్పరం మార్చాం. తిరిగి ప్రధాన ప్రోగ్రాంలో a, b విలువలను ప్రింట్ చేస్తున్నాం. ప్రోగ్రాంను రన్ చేసి చూడండి. a విలువ 10, b విలువ 20 అనే వస్తుంది. ప్రమేయంలో మనం a, b విలువలను ఒకదానికొకటి మార్చాం. అయితే ప్రధాన ప్రోగ్రాంలో ఈ మార్పు జరగలేదు ఎందువల్ల?

ఇంతకుముందు చూసినట్లు ఏ ప్రమేయంలో చరరాశుల విలువలు ఆ ప్రమేయానికే పరిమితం. అంటే చరరాశుల పేర్లు ప్రధాన ప్రోగ్రాంలోను, ప్రమేయంలోను ఒకటే అయినా అవి వేర్వేరు చరరాశుల కిందే లెక్క. ఒక దానిలో విలువ మార్పు మరోకదానిలో ప్రతిఫలించదు.

దీన్నిబట్టి చరరాశుల పేర్లు ఉపయోగించటం వల్ల కొన్ని ఇబ్బందులు ఉన్నాయని తెలిసింది. ఇదే మెమోరీలో చరరాశుల చిరునామాలను ఉపయోగించటం మనకు తేలికవుతుంది.

చరరాశుల చిరునామాలను కనుగొనటం ఎలా? కీబోర్డు సహాయంతో ఇన్‌పుట్ చేసిన విలువలను ఒక చరరాశికి ఇవ్వాలంటే scanf స్టేట్‌మెంట్‌ను ఉపయోగిస్తున్నాం. ఇందులో మనం చరరాశి పేరుకు ముందు '&' గుర్తును వాడుతున్నాం. ఈ గుర్తును అడ్రస్ ఆపరేటర్ అంటారు. దీన్ని ఉపయోగిస్తే, మనం ఎంటర్ చేసిన విలువ ఆ చిరునామా ఉన్న చరరాశికి చేరుతుందని అర్థం. ఇప్పుడు ఒక చరరాశి చిరునామా ఏవిధంగా ఉంటుందో పరిశీలిద్దాం.

```
#include<stdio.h>
main()
{
    int num1,
    longint*add;
    num = 10;
    add = &num;
    printf("The address of num is %u", add);
}
```

దీన్ని ఎగ్జిక్యూట్ చేయండి. మీకు num1 అనే చరరాశి చిరునామా కనిపిస్తుంది. ఈ చిరునామా విలువ int

పరిమితి కంటే ఎక్కువ ఉంటుంది. కాబట్టి మనం long int ని తీసుకున్నాం. ఇప్పుడు చిరునామా ఎలా ఉంటుందో మీకు అర్థమైందికదా! ఏ రెండు చరరాశులకు ఒకే చిరునామా ఉండదు.

ఈ చిరునామాలను మనం వేరే చరరాశులలో నిలవ ఉంచుకోవచ్చు. ఈ విధంగా చిరునామాలను నిలవ ఉంచడానికి ఉపయోగించే చరరాశులనే 'సి' పరిభాషలో పాయింటర్‌లు (pointers) అంటారు. పాయింటర్ తరహా చరరాశులు మెమొరీలోని ఒక ప్రాంతాన్ని దానిలో ఉండే విలువ, దాని పేరు బట్టికాక దాని చిరునామాను బట్టి సూచిస్తాయి. అయితే int, char, float వంటి వేర్వేరు డేటాటైప్‌లు వేర్వేరు పరిమాణాల మెమొరీని కలిగి ఉంటాయి. అందువల్ల వీటి చిరునామాలను సూచించే పాయింటర్లు కూడా వేర్వేరుగానే ఉంటాయి. అంటే int మెమొరీని సూచించే పాయింటర్లు, char ని సూచించే పాయింటర్లు, float ని సూచించే పాయింటర్లు విడివిడిగా ఉంటాయి. సాధారణంగా పాయింటర్ చరరాశులను డిక్లేర్ చేసేటప్పుడు ముందుగా డేటాటైప్‌ను ఇచ్చి తర్వాత ఒకే స్పేస్‌ను, తర్వాత చరరాశి పేరును ఇవ్వాలి. ఉదాహరణకి p అనే చరరాశి int కి సంబంధించిన పాయింటర్ అనుకుందాం. దాన్ని ఈ కింది విధంగా డిక్లేర్ చేయాలి.

```
int *p;
```

ఇందులో p అనేది చిరునామా చరరాశి (address variable) దీన్ని డిక్లేర్ చేయడానికి int * ని వాడతాం. అంటే int* అనేది డేటాటైప్‌లో భాగమేకాని చరరాశిలో భాగం కాదు. ఇప్పుడు char చరరాశి చిరునామాను నిలువ చేసేందుకు ఒక చరరాశిని డిక్లేర్ చెయ్యాలంటే

char *c; అని చెయ్యాలి.

అదే float కు

float *f అని డిక్లేర్ చేయవచ్చు.

ఒక చరరాశిని చిరునామా చరరాశిగా డిక్లేర్ చేసినప్పుడు దాని పేరును వేరేదానికి పెట్టకూడదు. డిక్లరేషన్ సమయంలో ఒక చిరునామా చరరాశికి, మరొక సాధారణ చరరాశిని డిక్లేర్ చేయవచ్చు. ఉదాహరణకి

```
int *p, num;
```

అని డిక్లేర్‌చేయటం సరైన పద్ధతే. ఇందులో p పాయింటర్‌కాగా num అనేది సాధారణ చరరాశి.

```
int x, y;

int *p;

p = &x;
```

పై స్టేట్‌మెంట్‌లో x అనే చరరాశి చిరునామాను p కి ఇస్తున్నాం. ఇందువల్ల p అనేది x ను సూచిస్తుంది. ఇప్పుడు x కి ఏదైనా విలువను ఇస్తే, p ద్వారా ఆ విలువను కనుక్కోవచ్చు. ఉదాహరణకి x కి 10 అనే విలువను ఇస్తే *p = 10 అవుతుంది. అంటే p అనేది x అనే చరరాశి చిరునామాను నిలవ ఉంచుకుంటే *p అనేది x యొక్క విలువను సూచిస్తుంది. ఈ కింది ఉదాహరణలో ఈ విషయాన్ని మరింత స్పష్టంగా చూద్దాం.

```
#include<stdio.h>

main()
```

```
{
    int *p;
    int num;
    p = &num;
    num = 10;
    printf("The variable *p points to value %d", *p);
    *p = 20;
    printf("The value in num charged to %d", num);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసినప్పుడు ఈ కింది విధంగా ప్రింటవుతోంది.

```
The variable *p points to value 10
The value of num chayed to 20
```

పై ప్రోగ్రాంలో num అనే చరరాశిని తీసుకుని దానికి 10 అనే విలవ ఇచ్చాం.

అంటే num

10

1024

ఇక్కడ num అనేది చరరాశి పేరు

10 అనేది చరరాశి విలువ

1024 చరరాశి చిరునామా

ఇప్పుడు చరరాశి చిరునామాను మరో చరరాశిలో భద్రపరుస్తున్నాం.

num	p	
10	1024	p = &num
1024	1048	

దీని తర్వాత 1024 అని చిరునామా ఉన్న మెమోరీలో విలువను మనం num ద్వారా కాని లేదా p ద్వారా కాని మార్చవచ్చు.

దీని బట్టి పైన ప్రమేయాన్ని ఉపయోగించిన ప్రోగ్రాంలో వచ్చిన ఇబ్బందిని తొలిగించే మార్గం చూద్దాం.

```
#include<stdio.h>
main()
```

```
int num, x, y;
void swap (int *a, int* b);
x= 10;
y = 20;
swap(&x, &y);
printf("The values of x,y are %d, %d,", x,y);
}
swap(int *d, int*b)
{
int * temp;
*temp = *a;
*a =*b;
*b= *temp;
}
```

దీన్ని ఎగ్జిక్యూట్ చేస్తే ఫలితం ఈ కింది విధంగా వస్తుంది.

The values of x, y are 20, 10

ఇప్పుడు ప్రధాన ప్రోగ్రాంలో విలువలు మారిపోతాయి చూశారా! దీనికి కారణం మనం విలువలకు బదులుగా వీటి చిరునామాలను ప్రమేయంలోకి పంపాం. కింద ఆ చిరునామాల్లోని విలువలను మనం పరస్పరం మార్చాం. అందువల్ల ప్రధాన ప్రోగ్రాంలో x, y ల విలువలుకూడా పరస్పరం మారాయి.

శ్రేణులు-పాయింటర్లు

శ్రేణులకి, పాయింటర్‌లకు అవినాభావ సంబంధం ఉంది. నిజానికి శ్రేణి పేరే ఒక చిరునామా చరరాశి. ఉదాహరణకి

	1024	1026	1028	1030	1032	1034	1036
mark	40	50	90	60	75	80	55

ఈ శ్రేణిని మనం mark[7] అని సూచిస్తాం. అయితే ఇందులో mark అనే పేరు శ్రేణిలో మొదటి చరరాశి చిరునామాను సూచిస్తుంది. ఇప్పుడు శ్రేణిలో ఉన్న చరరాశుల చిరునామాలను పైన రాశాం. mark అనే చరరాశిలో 1024 అన విలువ ఉంటుంది.

mark = 1024

*mark = 40

రెండో విలువను అంటే 1026 చిరునామాగా గల 50 ని సూచిస్తుంది. ఇదే *(mark+5) 55ని సూచిస్తుంది.

పైన చూపిన ఉదాహరణలో mark అనేది శ్రేణిలోని మొదటి మెమొరీ స్థానం చిరునామా అని తెలుసుకున్నాం కదా! ఇప్పుడు మిగిలిన మెమొరీ స్థానాల చిరునామాలు ఏవిధంగా ఉంటాయో చూద్దాం. శ్రేణిలోని సూచీ విలువ మారుతూఉంటే శ్రేణి విలువ కూడా మారుతూ ఉంటుంది.

సూచీ విలువ	శ్రేణి	పాయింటర్	చిరునామా
0	mark[0]=40	*(mark+0)=40	&mark[0] లేదా mark+0=1024
1	mark[1]=50	*(mark+1)=50	&mark[1] లేదా mark+1=1026
2	mark[2]=90	*(mark+2)=90	&mark[2] లేదా mark+2=1028
3	mark[3]=60	*(mark+3)=60	&mark[3] లేదా mark+3=1030
4	mark[4]=75	*(mark+4)=75	&mark[4] లేదా mark+4=1032
5	mark[5]=80	*(mark+5)=80	&mark[5] లేదా mark+5=1034
6	mark[6]=55	*(mark+6)=55	&mark[6] లేదా mark+6=1036

పైన చూపినట్లుగా శ్రేణిలోని ప్రతి మూలకానికి ఒక చిరునామా ఉంటుంది. ఇప్పుడు శ్రేణిని ఎన్ని విధాలుగా వ్యక్తీకరించవచ్చు చూశాం.

మామూలు శ్రేణిని

mark[5]=80 అని సూచించవచ్చు.

అదే పాయింటర్‌ను ఉపయోగించి

*(mark+5) అని సూచించవచ్చు.

ఇక్కడ *(mark+5) అంటే ప్రస్తుతం మనకు కావలసిన సంఖ్య mark కి 5 సంఖ్యల దూరంలో ఉందని అర్థం. ఒక్కో సంఖ్యకి మెమొరీలో రెండు బైట్లు (Bytes) కేటాయించబడతాయి. కాబట్టి ప్రస్తుత సంఖ్య mark కి 2×5=10 బైట్‌ల దూరంలో ఉంటుంది. అందుకనే mark చిరునామా 10124 అయితే mark+5 చిరునామా 1034 అవుతుంది.

పైన చూపినదాన్నిబట్టి మనకు ఒక విషయం చాలా స్పష్టంగా తెలుస్తోంది. ఏదైనా శ్రేణిలో మొదటి మూలకం తెలిస్తే (మూలకం పేరే చిరునామాను సూచిస్తుంది కాబట్టి) చాలు మిగిలిన మూలకాలను చాలా తేలిగ్గా కనుక్కోవచ్చు.

ఇప్పుడు ప్రమేయాలను మళ్లీ పరిశీలిద్దాం. ప్రమేయాలలోకి విలువలను ఒక్కొక్కటే ఇవ్వగలం. అంటే ప్రధాన ప్రమేయం (main) నుంచి ఇతర ప్రమేయాలకు విలువలను పంపేటప్పుడు శ్రేణి మొత్తాన్ని ఒక సారే పంపలేము. అయితే పాయింటర్‌లను ఉపయోగించినపగపడు శ్రేణిపేరును లేదా శ్రేణి మొదటి మూలకం చిరునామాను పంపితే సరిపోతుంది. దాని ద్వారా మిగిలిన విలువలను కూడా గ్రహించవచ్చు. కాని శ్రేణి పేరుతోపాటు ఆశ్రేణిలో మూలకాలు ఎన్ని ఉన్నాయో కూడా మనం పంపాలి. ఈ కింది ఉదాహరణను పరిశీలించండి

```
#include<stdio.h.
#include<conio.h>
main()
{
    int i, num[10];
    void print_num (int *n, int k);
    for(i=0; i<=9; i++)
    {
        printf("Enter a number \n");
        scanf("%d", &num[i]);
        clrscr();
    }
        printf_num(num, 10);
    }
void print_num (int *n, int k)
{
    int j;
    for (j=0; j<k; j++)
    {
        printf("The num is %d", *(n+j));
    }
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

ఈ ప్రోగ్రాంలో మనం ప్రమేయాన్ని డిక్లేర్ చేసేటప్పుడు int *n అన్న చరరాశిని డిక్లేర్ చేశాం. విలువను పంపేటప్పుడు శ్రేణి మొదటి మూలకం చిరునామాను సూచించే num ను పంపాం. అదేవిధంగా రెండో int వద్ద నేరుగా 10ని పంపాం. కింద n ను ఉపయోగించి j విలువను పెంచటం ద్వారా శ్రేణిలో మూలకాలను ప్రింట్ చేయగలుతుతూన్నం. ఇదే విధంగా మనం పదాలను, వాక్యాలను కూడా ప్రమేయాలకు పంపవచ్చు. దీనికి సంబంధించిన ఉదాహరణను కింద చూద్దాం.

✦✦✦

అధ్యాయం 11

చిత్రాలు (Structures)

మెమొరీలోని వివరాలను సాధారణంగా iny, float, char, double వంటి డేటాటైప్‌లు ఉపయోగించి వర్గీకరిస్తాం. అంటే మన వివరాలను ఆడేటా విలువలుగా భద్రపరుస్తాం. మనకు కావలసిన వివరాలన్నీ ఈ డేటాటైప్‌ల సహాయంతో వ్యక్తపరచవచ్చు. అయితే కొన్ని రకాల వివరాలను భద్రపరచాలంటే ఈ డేటాటైప్‌లు కొంత అయోమయాన్ని సృష్టించవచ్చు. ఉదామరణకి పదిమంది ఉద్యోగుల పేర్లు, విభాగాలు, వేతనాలవంటి వివరాలను భద్రపరచాలనుకుందాం. వాటికి చరరాశులను డిక్లేర్ చేస్తే అవి ఈ కింది విధంగా ఉంటాయి.

```
char name[10][20];

char dept[10][4];

float salary[10];
```

పై వివరాలను ఒక్కొక్కటిగా ఇవ్వాలంటే ఈ మూడు చరరాశులకు ఒకే పద్ధతిలో విలువలను ఇవ్వాలి. ఏ ఉద్యోగికి మనం ఏ విభాగాన్ని ఇస్తున్నామో, ఎంత వేతనం ఇస్తున్నామో తెలియజేయాల్సి ఉంటుంది. ఈ వివరాలను క్రమపద్ధతిలో ఉంచటం చాలా కష్టమైన పని, తప్పులు జరగటానికి ఆస్కారం ఎక్కువ.

అదే •

```
char name[20];
char dept [4];
float salary;
```

అని ఒక ఉద్యోగికి సంబంధించిన వివరాలను ఒక చిత్రంలో బిగించి అటువంటి చిత్రాలను ప్రతి ఉద్యోగికి ఒక్కొక్కటి కేటాయిస్తే సరిపోతుంది కదా? ఈ రకమైన వివరాలను నిలవ చేయటం కోసం 'సి' భాష మనకు చిత్రాలను (Structures) అందిస్తోంది. ఒక ఉద్యోగికి సంబంధించిన వివరాలను ఒక చిత్రంలో ఉంచి, అటువంటి చిత్రాలను అనేకం మనం ఉపయోగించుకోవచ్చు. STRUCTURE ను ఈ కింది విధంగా నిర్వచించవచ్చు.

```
struct
{
        char name[20];
        char dept[4];
        float salary;
}
```

ఇందులో struct అనేది సి భాషకు చెందిన Reserved Word. ఆ తర్వాత బ్రాకెట్‌లు తప్పనిసరిగా ఉండాలి. బ్రాకెట్ల లోపల మనకు కావలసిన చరరాశులను డిక్లేర్ చేయాలి. ఈ చరరాశులను structure లో సభ్యులు (members) అంటారు. పై structure లో name, dept, salary అనేవి సభ్యులవుతాయి. ప్రతి చట్రంలో ప్రతి సభ్య చరరాశికి విడివిడిగా పేర్లుండాలి. ఇప్పుడు మనం ఈ మొత్తం చట్రానికి ఒక పేరు పెట్టాలి. చట్రం మొత్తాన్ని ఒక చరరాశిగా వ్యవహరించాలంటే దానికి ఒక పేరుండాలి. ఈ చట్రం చరరాశి (structure variable)ని కింది విధంగా డిక్లేర్ చేయవచ్చు.

```
struct
{
    char name[20];
    char dept[4];
    float salary;
}
    employee;
```

ఇక్కడ employee అనేది చరరాశి. బ్రాకెట్ ముగించిన తర్వాతే ఈ చరరాశి పేరును ప్రకటించాలి. ఇప్పుడు employee అనేది చట్రానికి సంబంధించిన చరరాశి. దీనిలో 3 సభ్యచరరాశులున్నాయి.

చట్రాలకు విలువలను ఇవ్వటం: చట్రానికి విలువలను ఇవ్వటం అంటే చట్రంలోని చరరాశులకు విలువలను ఇవ్వటమే. ఈ చట్రంలోని చరరాశులను మనం నేరుగా ఉపయోగించటానికి వీలులేదు. చట్ర సహాయంతో సభ్యచరరాశులను వాడుకోవచ్చు. ఉదాహరణకి పై చట్రంలో salary అనే చరరాశికి విలువలను ఇవ్వాలంటే

```
employee.salary=4250.50;
```

అని ఇవ్వవచ్చు.

ఇక్కడ employee కి salary కి మధ్య ఒక చుక్క (period) ను ఉపయోగించాము. దీన్ని పీరియడ్ ఆపరేటర్ అంటారు. చట్రచరరాశికి, అందులోని సభ్యచరరాశిని అనుసంధానించటానికి ఈ ఆపరేటర్ (చుక్క) ఉపయోగపడుతుంది. ఈ విధంగా చట్రంలోని సభ్యచరరాశులను మనం మామూలు చరరాశులలాగానే ఉపయోగించవచ్చు. ఈ కింది ఉదాహరణతో చట్రాలను పూర్తిగా అర్థం చేసుకోవచ్చు.

```
#include<stdio.h.
#include<conio.h>
struct
{
    int emp_no;
    char ename[20];
    float salary;
```

```
        }
      emp_1, emp_2; /*structure variables related to two employees */
main()
{
   int i;
   printf("Enter details of first employee");
   printf("\n Enter employee number \n");
   scanf("%d", &emp_1.emp_no);
   printf("Enter name \n");
   scanf("%s", emp_1.ename);
   printf("Enter salary \n");
   scanf("%f", &emp_1.salary);
   clrscr();
   printf("Enter details of second employee\n");
   printf("Enter employee number \n");
   scanf("%d", &emp_2.emp_no);
   printf("Enter employee name \n");
   scanf("%s", emp_2.ename);
   printf("Enter salary \n");
   scanf("%f", &emp2.salary \n");
   clrscr();
   printf("The details of first employee are \n");
   printf("Employee no, name and salary are \n");
   printf(%d \n %s \n %f", emp_1. emp_no, emp_1.ename, emp_1.salary);
   printf("\n Enter any key to continue \n");
   getch();
   printf("The details of second employee are \n");
   printf("No: %d \n Name: %s \n Salary: %f", emp_2.emp_no, emp_2.ename, emp_2.salary);
}
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

Enter details of first employee

Enter employee number

10

Enter name

Ramu

Enter salary

2860

Enter details of Second employee

Enter employee number

20

Enter employee name

Kiran

Enter Salary

4280

The details of first employee are

Employee No, Name and Salary are

10

Ramu

2860

Press any key to continue

The details of second employee are

No: 20

Name : Kiran

Salary: 4280

పై ఉదాహరణలో రెండు చత్రచరరాశులను తీసుకున్నాం. ఈ రెండింటికీ సభ్యచరరాశులు ఒకటే గమనించండి. అయితే ప్రతి చత్రచరరాశికీ ఈ మూడు సభ్య చరరాశుల మొత్తం మెమొరీ కేటాయించబడుతుంది. సాధారణంగా int కి 2 బైట్‌లు, char కి 1 బైట్, float కి 4 బైట్‌లు మెమొరీ కేటాయించబడుతుంది. అయితే చత్ర చరరాశికి నిర్దిష్ట మొత్తం మెమొరీ కేటాయించబడుతుంది. అయితే చత్ర చరరాశికి నిర్దిష్ట మొత్తం మెమొరీ అంట్యూ ఉండదు. ఆ చత్రంలోని చరరాశుల మొత్తం మెమొరీ చత్రచరరాశి మెమొరీగా కేటాయించబడుతుంది. అంటే ఎంత మెమొరీ కావాలో మనమే నిర్ణయించుకోవచ్చు. అందుకనే చత్రాలను (structures ను) user defined data types అంటారు. పైన పేర్కొన్న emp_1, emp_2 చరరాశులు ఎంత మెమొరీని తీసుకుంటాయో తెలుసుకోవాలంటే అందులోని చరరాశుల మెమొరీ మొత్తాన్ని కూడాలి.

int emp_no అనే చరరాశి 2 బైట్‌లను తీసుకుంటుంది.

char name[20] అనే శ్రేణిలో 20 బైట్‌లుంటాయి.

float salary అనే చరరాశికి 4 బైట్‌లు కేటాయించబడతాయి

కాబట్టి మొత్తం చట్రానికి 2+20+4 = 26 బైట్‌ల మెమొరీ కావాలి. అంటే emp_1, emp_2 చరరాశులకి ఒక్కోదానికి 26 బైట్‌ల మెమొరీ కేటాయించబడుతుంది. ఇదే విధంగా చట్ర చరరాశుల మెమొరీని లెక్క కట్టవచ్చు.

చట్రాలు-శ్రేణులు

సాధారణ చరరాశులకి శ్రేణులు ఉన్నట్లు చట్రాల చరరాశులకు కూడా శ్రేణులుండవచ్చు. వీటిని కూడా మామూలు శ్రేణుల లాగా డిక్లేర్ చేయవచ్చు. ఉదాహరణకి పై ప్రోగ్రాంలో చరరాశులను పదిమంది ఉద్యోగులకు చెందేలా రాయాలంటే

```
struct
{
    int emp_no;
    char name[20];
    float salary;
} emp[10];
```

అని డిక్లేర్ చేయవచ్చు. దీనికి సంబంధించిన ఒక ప్రోగ్రాం ఇప్పుడు చూద్దాం.

ఈ ప్రోగ్రాంలో ఒక బ్యాంక్ అకౌంట్ నంబరు, పేరు, డబ్బు మొత్తం అని మూడు చరరాశులు తీసుకుందాం. ఇందులో డిపాజిట్ మొత్తం ప్రతి వెయ్యి రూపాయాలకి ఒక వందరూపాయల వడ్డీ కలిపి మళ్లీ దాన్ని డిపాజిట్ చెయ్యాలి.

```
#include<stdio.h>
#include<conio h>
struct
{
    int ac_no;
    char name[20];
    float amount;
} acc[10];
main()
{
    int i;
    float interest;
```

```
for(i=0; i<=9; i++)
{
    printf("Eter ac number \n");
    scanf("%d", &acc[i].ac_no);
    printf("Enter name \n");
    scanf("%s", acc[i].name);
    printf("Enter amount n");
    scanf("%f", &acc[i].amount);
}
/* Now we calculate interest */
for(i=0; i<=9; i++)
{
    interest = (acc[i].amount/1000)*100;
    acc[i].amount=acc[i].amount+interest;
}
/* display the details */
for(i=0; i<=9; i++)
{
    printf("Acc_No: %d", acc[i].ac_no);
    printf("\n Name: %s \n", acc[i].name);
    printf("\n Amount: %f \n", acc[i].amount);
}
}
```

పై ప్రోగ్రాంను ఎగ్జిక్యూట్ చేసి ఫలితం చూడండి.

ఈ విధంగా మనం చిత్రాల శ్రేణుల సహాయంతో కొన్ని పూర్తి ప్రోగ్రాములు రాయవచ్చు.

ఇదే విధంగా మనం పాయింటర్లను కూడా ఉపయోగించి చిత్రాలను వ్యక్తీకరించవచ్చు.

పాయింటర్‌లను ఈ కింది విధంగా డిక్లేర్ చేయవచ్చు.

```
struct
{
  int no;
  char name[20];
  } *emp;
```

సభ్య చరరాశులను అనుసంధించాలంటే Arrow Operator ను ఉపయోగిస్తాం.

```
emp→no;
emp→name;
```

అని వ్యక్తం చేయవచ్చు.

కింది ఉదాహరణను పరిశీలించండి.

```
#include<stdio.h>
#include<conio.h>
struct
{
    int no;
    char name[20];
} #emp;
main()
{
    emp→no = 40;
    printf("Enter name \n");
    scanf("%s", emp→name);
    clrscr();
    printf("The details you etered are \n");
    printf("Emp number is %d \n", emp→no);
    printf("Name is %s \n", emp→name);
}
```

ఈ ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే పాయింటర్‌లు ఎలా పనిచేస్తాయో తెలుస్తుంది.

✦ ✦ ✦

అధ్యాయం 12

'సి'లో ఫైళ్ల కార్యకలాపాలు
File Operations in 'C'

కంప్యూటర్లలోకి మనం వివరాలను ఇన్‌పుట్‌గా ఇస్తున్నాం. ఫలితాలను ఔట్‌పుట్‌గా తీసుకుంటున్నాం. వీటితో పాటు మన వివరాలను భద్రపరిచే ప్రక్రియలను కూడా నిత్యం చేస్తూనే ఉన్నాం. వివరాలను తాత్కాలికంగా భద్రపరచాలంటే వాటిని కంప్యూటర్ ప్రధాన (primary) మెమొరీలో ఉంచుతాం. అదే శాశ్వతంగా భద్రపరచాలంటే వాటిని ఫ్లాపీలు, హార్డ్‌డిస్క్ వంటి పరిసరాలలో భద్రపరుస్తాం.

మెమొరీ అని మనం ఇప్పటి వరకూ ఉపయోగించింది తాత్కాలిక మెమొరీనే. ఫ్లాపీడిస్క్, హార్డ్‌డిస్క్ వంటి పరికరాలను బాహ్యాపరికరాలు (External Devices) అంటారు. 'సి' ప్రోగ్రామ్‌లలో ఇప్పటి వరకూ మనం ఉపయోగించిన చరరాశులన్నీ తాత్కాలిక మెమొరీని మాత్రమే ఉపయోగించుకున్నాయి. అందువల్లనే ప్రోగ్రాం రన్ అవటం పూర్తవగానే ఈ చరరాశులు మాయమవుతాయి. అంటే చరరాశుల జీవితకాలం కేవలం ప్రోగ్రాం రన్ అవుతున్నంత సేపు మాత్రమే. ప్రోగ్రాంలలోకి మనం ఎంటర్ చేసిన డేటా, ప్రోగ్రాం పూర్తికాగానే అదృశ్యమవుతుంది. అయితే మన వివరాలను శాశ్వతంగా భద్రపరచలేమా? అదీ సాధ్యమే. మనం ఎంటర్ చేసిన వివరాలను శాశ్వతంగా భద్రపరచాలంటే వాటిని బాహ్యాపరికరాలలైన ఫ్లాపీడిస్క్, హార్డ్‌డిస్క్ వంటి వీటిలో ఉంచాలి. ఈ పరికరాలలో వివరాలను భద్రపరచాలంటే మన డేటాను ఫైళ్ల రూపంలో భద్రపరచాలి. ఈ ఫైళ్లను సృష్టించటం, వాటిలోకి వివరాలను రాయటం, ఆ వివరానలు మార్చటం, అనవసరమైన వాటిని తొలగించటం, అవసరం పూర్తయిన తర్వాత ఆ ఫైళ్లను తొలగించటం (Delete చేయటం) వంటి కార్యకలాపాలను 'సి' లో మనం సమర్థంగా నిర్వహించవచ్చు.

కొన్ని వివరాల సముదాయాన్నే ఫైల్ అంటారు. మనం ఒక ప్రోగ్రాం రాసి దాన్ని ఒక ఫైల్‌గా భద్రపరుస్తున్నాం. ఆ ఫైల్‌లో మన ప్రోగ్రాం ఉంటుంది. అదే విధంగా ఎంఎస్‌వర్డ్ ఉపయోగించి కొన్ని లేఖలు, వివరాలను ఫైల్‌గా భద్రపరుస్తాం. ఫైల్‌లో భద్రపరిచే వివరాలు ఏవైనా కావచ్చు. ఒక అక్షరం నుండి కొన్ని పేరాలు, పేజీలు, పుస్తకాల వరకూ కూడా ఒక ఫైల్‌లో భద్రపరచవచ్చు. ఒక ఫైల్ మెమొరీలో ఎంతస్థానాన్ని ఆక్రమిస్తుందో దాన్ని ఆ ఫైల్ పరిమాణం అంటారు. ఫైల్ పరిమాణం ఇంతే ఉండాలన్న నిబంధన ఏదీలేదు. అది ఒక బైట్ నుంచి కొన్ని మెగాబైట్‌ల వరకూ ఉండవచ్చు. అయితే ఒక నిర్దిష్ట ఫైల్ పరిమాణం ఎంత ఉండాలనే అంశం మాత్రం మనం ఉపయోగించే ఆపరేటింగ్ సిస్టమ్‌లపై ఆధారపడి ఉంటుంది.

'సి' భాషను ఉపయోగించి ఫైళ్లను సృష్టించటం, వాటితో వివిధ కార్యకలాపాలు జరిపే ప్రక్రియలను మనం ఇప్పుడు పరిశీలిద్దాం. బాహ్య పరికరాలైన డిస్క్, కీబోర్డ్, మానిటర్‌లతో మనం జరిపే కార్యకలాపాలన్న ఫైళ్ల సహాయంతోనే జరుగుతాయి. అంటే మానిటర్ పై ఏదైనా ప్రింట్ చేయాలంటే ఆ వివరాలను మనం ఒక ఫైల్‌లోకి రాస్తాం. అదే ఏవైనా ివరాలను తీసుకోవాలంటే వాటిని వేరే ఫైల్ నుంచి తీసుకుంటాం. మరింత వివరంగా చెప్పాలంటే, సి భాష standard utput అనే ఫైలును, standard input అనే ఫైలును ఉపయోగించుకుంటుంది. మానిటర్ పై ఏవైనా వివరాలను ప్రింట్

చేయాలంటే putchar, printf ల సహాయంతో standard output ఫైల్‌లోకి రాస్తాం. అదే ఏవైనా వివరాలను ప్రోగ్రాంలోకి తీసుకోవాలంటే standard input ఫైలులోంచి scanf లేదా getchar వంటి ప్రమేయాల సహాయంతో తీసుకుంటుంది. దీన్ని బట్టి ఫైల్‌లోకి వివరాలను రాయాలంటే printf, putchar వంటి ప్రమేయాలను, ఫైల్ నుంచి తీసుకోవాలంటే getchar వంటి ప్రమేయాలను ఉపయోగించాలని తెలుస్తోంది. getchar ఉపయోగించి విలువలను తీసుకోవటం చూశాం అదే విధంగా putchar ఉపయోగించి విలువలను ప్రింట్‌చేయటం ఎలాగో చూద్దాం.

```
#include<stdio.h>
main()
{
    putchar('L');
    putchar('I');
    putchar('O');
    putchar('N');
    putchar('\n');
}
```

దీన్ని ఎగ్జిక్యూట్ చేస్తే

LION అని ప్రింటవుతుంది.

ఈ విధంగా putchar ఉపయోగించి ఒక్కో విలువను ప్రింట్ చేయవచ్చు. ఒక్కో విలువను ప్రింట్ చేయటం ద్వారా మనం ప్రతి అక్షరంపై నియంత్రణను సాధించవచ్చు. పై ప్రోగ్రాంలో standard input, standard output ఫైళ్లను ఉపయోగించాం. కాని వాటిని include చేయలేముకదా అన్న అనుమానం మీకురావచ్చు. నిజానికి stdio.h అనే ఫైల్‌లో పైరెండూ ఫైళ్లను include చేస్తాం అందువల్ల ఇక్కడ వాటిగురించి పట్టించుకోనక్కరలేదు.

ఫైల్‌ని డిక్లేర్ చేయటం:

'సి' భాషలో ఫైల్‌ని కూడా ఒక డేటాటైప్‌గా పరిగణించి దానికి చరరాశులను ప్రకటిస్తాం. ఈ ఫైల్‌ని డిక్లేర్ చేయటానికి FILE ని ఉపయోగిస్తాం. మిగిలిన పదాలన్నీ చిన్న అక్షరాలనే ఉపయోగించినా FILE ను మాత్రం పెద్ద అక్షరాలు (Upper case) లోనే వ్యవహరిస్తాం. దీనికి ఒక కారణం ఉంది. FILE అనేది నిజానికి 'సి' భాషలో అసలైన డేటాటైప్ కాదు. దీన్ని stdio.h అనే హెడర్ ఫైల్‌లో నిర్వచించారు. 'సి'లో సిసలైన డేటాటైప్‌లకు, దీనికి మధ్య తేడా చూపటం కోసం FILE ను పెద్ద అక్షరాలతో సూచిస్తారు. ఫైల్ చరరాశులను ఈ కింది విధంగా డిక్లేర్ చేయవచ్చు.

```
FILE *fp;
```

ఇక్కడ fp అనేది సాధారణ చరరాశి. దీన్ని File pointer చరరాశి అంటారు. దీన్నే ఫైల్ చరరాశిగా వ్యవహరించవచ్చు. మెమొరీలో ఒక ఫైల్‌ను ఇది సూచిస్తుంది. ఇంతకుముందు చెప్పినట్లుగా ఒక ఫైల్‌ను కొత్తగా సృష్టించవచ్చు, దీనిలోకి వివరాలను రాయవచ్చు. అందులోంచి వివరాలను గ్రహించవచ్చు, ఉన్న వివరాలకు కొత్త వివరాలను కూడా జత చేయవచ్చు.

ఒక ఫైల్‌ను డిక్లేర్ చేయటమంటే దాన్ని సృష్టించినట్లే. ఇక ఫైల్‌లో ఏమన్నా రాయాలన్నా, మిగిలిన ప్రక్రియలు జరపాలన్నా ఆఫైల్‌ను ముందుగా ఓపెన్ చేయాలి. ఒక ఫైల్‌ను ఓపెన్ చేస్తేనే అందులో రాయటం, వివరాలను తీసుకోవటం, కొత్త వాటిని జతచేయటం వంటి ప్రక్రియలకు వీలుంటుంది. ప్రోగ్రాం పూర్తవగానే ఫైల్‌ను క్లోజ్ చేయాలి. ఈ విధంగా ఓపెన్ చేసిన ప్రతి ఫైల్‌ను తప్పనిసరిగా క్లోజ్ చెయ్యాలి. ఫైల్ ఓపెన్ చేయటానికి fopen అనే ప్రమేయం ఉపయోగపడుతుంది. అదే క్లోజ్ చేయటానికి fclose అనే ప్రమేయాన్ని వాడతారు. ఒక ప్రోగ్రదంలో ఒక ఫైల్‌ను ఎన్నిసార్లయినా ఓపెన్ చేసి క్లోజ్ చేయవచ్చు. అయితే ఒక సారి ఓపెన్ చేసిన ఫైల్‌ను క్లోజ్ చేస్తేనే గాని మళ్లీ ఓపెన్ చేసే వీలుండదు.

ఫైల్‌పై ఏ ప్రక్రియలు జరపాలన్నది ఫైల్‌ను ఏవిధంగా ఓపెన్ చేశామనే దానిపై ఆధారపడి ఉంటుంది. అంటే ఫైల్‌లోకి ఏమన్నా రాయాలంటే ఒక విధంగాను, దాని నుంచి ఏమన్నా తీసుకోవాలంటే మరో విధంగాను, అప్పటికే ఉన్న వివరాలకు కొత్త వాటినిన జత చేయాలంటే మరో విధంగాను ఫైల్‌ను ఓపెన్ చేయాల్సి ఉంటుంది. దీన్నే ఫైల్ ఓపెన్ చేసే విధానం (mode) అంటారు. ఈ విధానం రాసే విధానమయితే (writing mode) ఆఫైళ్లలో కేవలం కొత్త వివరాలను రాయటానికి మాత్రమే వీలుంటుంది. చదివే వీలుండదు. అదే ఫైల్‌ను చదివే విధానం (reading mode) లో ఓపెన్ చేస్తే అప్పుడు కేవలం ఫైల్‌లో వివరాలు చదవటం మాత్రమే కుదురుతుంది. అదే append mode లో ఓపెన్ చేసే పాత వివరాలను చదివి కొత్త వివరానలు రాయవచ్చు.

fopen ప్రమేయం-ఈ ప్రమేయాన్ని కింది విధంగా ఓపెన్ చేయవచ్చు.

```
FILE *fp;

fp = fopen("Datafile.dat", "w");
```

ఇక్కడ fp అనేది ఫైల్ సూచి

fopen ప్రమేయంలో రెండు భాగాలున్నాయి. మొదటి భాగంలో Datafile.dat అని రాశాం. ఇది ఫైల్ పేరుని సూచిస్తుంది. మనం ఫ్లాపీలో లేదా హార్డ్‌డిస్క్‌లో ఈ పేరుతోనే ఫైల్‌నుఏ భద్రపరుస్తాం. రెండో భాగంలో w అని రాశాం. ఫైల్ ఓపెన్ చేసే విధానం (mode) ను ఇదిసూచిస్తుంది. w అంటే writing mode అని అర్థం. అక్కడ r అని రాస్తే reading mode, a అని రాస్తే append mode అవుతుంది. మొదట భాగంలోని ఫైల్ పేరును మనం ఏదైనా పెట్టవచ్చు. ఫైల్‌ను w ఉపయోగించి రాసే విధానంలో ఓపెన్ చేస్తే, ప్రతి సారీ ఒక కొత్త ఫైల్ సృష్టించబడుతుంది. ఒకవేళ అదే పరుతో ఇంకోఫైల్ ఉన్నప్పటికీ ఆఫైల్ స్థానంలో కొత్త ఫైల్ సృష్టించబడుతుంది. ఏదైనా ఫైల్‌ను r ఉపయోగించి చదివే విధానం (reading mode) లో ఓపెన్ చేయాలంటే ఆఫైల్ అప్పటికే మన హార్డ్‌డిస్క్ లేదా ఫ్లాపీలో ఉండి ఉండాలి. ఒకవేళ ఆఫైల్ లేకపోతే ఎర్రర్ వస్తుంది. అదే ఒక ఫైల్‌ను a ఉపయోగించి append mode లో ఓపెన్ చేస్తే - ఆ ఫైల్ గనుక అప్పటికే డిస్క్‌లో ఉంటే అదే ఓపెన్ అవుతుంది. ఒకవేల లేకపోతే కొత్త ఫైల్ సృష్టించబడుతుంది. fopen ఉపయోగిస్తే, మనఫైల్‌ను fp అనే ఫైల్ పాయింటర్‌తో సూచిస్తున్నట్లువుతుంది. మన ప్రోగ్రాంలో ఫైల్‌ను దాని అసలు పేరుతో కాక కేవలం ఫైల్ పాయింటర్‌తో మాత్రమే వ్యవహరిస్తాం. ఒక ఫైల్‌ను ఎన్ని పాయింటర్‌లతోనైనా సూచించవచ్చు. అయితే ఒక పాయింటర్ కేవలం ఒక ఫైల్‌ను మాత్రమే సూచించవచ్చు. ఫైల్‌ను క్లోజ్ చేస్తిన తర్వాత ఆ పాయింటర్ విడుదలవుతుంది. అప్పుడు ఆ పాయింటర్‌ను మరో ఫైల్‌ను సూచించటానికి ఉపయోగించవచ్చు.

ఫైళ్లను క్లోజ్ చేయటానికి fclose అనే ప్రమేయాన్ని వాడతారు. ఈ fclose ను కింది విధంగా ఉపయోగించవచ్చు.

fclose(fp);

ఏఫైల్‌ను క్లోజ్ చేయదలచుకున్నామో దాని ఫైల్ పాయింటర్‌ను ప్రమేయం బ్రాకెట్‌లో ఇవ్వాలి.

ఫైల్ ముగింపు: ప్రతిఫైల్ ముగియటానికి ఒక గుర్తు ఉంటుంది. దాన్ని ctrl-zen చూసిస్తాం. దీన్నే EOF అని ప్రోగ్రాంలో సూచించవచ్చు. అంటే ఏవైనా వివరాలను మనం ఇస్తున్నప్పుడు control కీని, z అక్షరం ఉన్న కీని ఒకేసారి నొక్కితే EOF వస్తుంది. ఈ కింది ఉదాహరణను గమనించండి.

```
#include<stdio.h>
#include<conio.h>
main()
{
    FILE *fopen;
    char c;
    /* open the file using fopen */
    fp = fopen("Hello.data", "w");
    /* file opened in writing mode */
    printf("Enter text \n");
    while((c=getch())!=EOF)
    putc(c, fp);
    fclose(fp); /*close the file */
    /* open the file again to read */
    fp = fopen("Hello.dat", "r");
    printf("You stored the following in Hello.dat \n");
    while((c=getc(fp))!=EOF)
    putchar(c);
    fclose(fp); /* close again */
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే

Enter Text

Good morning

How are you?

ఇప్పుడు control, z అనే కీలను ఒకేసారి నొక్కితే EOF వస్తుంది. తర్వాత

You stored the following in Hello.dat

Good morning

How are you?

అని ఔట్‌పుట్ వస్తుంది.

పై ప్రోగ్రాంలో ముందుగా getch సహాయంతో EOF వచ్చే వరకూ అక్షరాలను, గుర్తులను ఇన్‌పుట్‌గా తీసుకున్నాం. ఈ ఇన్‌పుట్‌ను c అనే చరరాశిలో నిలవ ఉంచాం. అదే putc అనే ప్రమేయంలో c లోని విలువను fp ద్వారా ఫైల్‌లో నిలవ ఉంచాం.

రెండోసారి ఫైల్‌ను ఓపెన్ చేసినప్పుడు getc అనే ప్రమేయంలో fp ద్వారా ఫైల్‌లోని అక్షరాలను, గుర్తులను ఒక్కొక్కటిగా c లోకి తీసుకుంటున్నాం. తర్వాత putchar ద్వారా వాటిని తెరపై ప్రింట్ చేస్తున్నాం.

ఇక్కడ ఒక విషయం గమనించాలి. మనం కీబోర్డు ద్వారా ఎంటర్ చేసే ప్రతి గుర్తు ఫైల్‌లో నిలవ ఉంటుంది. అక్షరాల మధ్యలో స్పేస్‌లు, ఆలైన్ చివర ఉన్న Newline character ("\n") కూడా ఫైల్‌లో నిలవ ఉంటాయి. EOF వచ్చే వరకూ మనం ప్రింట్ చేయమని చెప్పాం కాబట్టి మొత్తం తిరిగి తెరపై కనబడుతుంది. ఇప్పుడు ఒకసారి DOS లోకి వచ్చి చూడండి Hello.dat అనే ఫైల్ మీడైరెక్టరీలోకి కొత్తగా వచ్చి ఉంటుంది.

మరో ఫైల్‌ను సృష్టించి ఈ ఫైల్ వివరాలు రెండో ఫైల్‌లోకి Copy చేసే మరో ప్రోగ్రాం ఇప్పుడు రాద్దాం.

```
#include<stdio.h>
#include<conio.h>
main()
{
    FILE *fopen();
        *fp_1;
        *pf_2;
    char c;
    fp-1=fopen("Hello.dat", "r");
    /* old file opened for reading */
    fp_2=fopen("second.dat", "w");
    /* second file opened in writing mode */
    while(c=getc(fp_1)!=EOF)
    putc(c, fp_2);
    /* close both files */
    fclose(fp_1);
```

```
fclose(fp_2);
/* Now open second file for reading */
fp_2=fopen("second.dat", "r");
while(c=getc(fp_2)!=EOF)
putchar(c);
fclose(fp_2);
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే

Good morning

How are you?

అని ప్రింటవుతుంది.

ఈ ప్రోగ్రాంను పరిశీలించండి. ముందు మనం Hello.dat అనే ఫైల్‌లో వివరాలను ఒక్కో అక్షరమే second.dat అనే ఫైల్‌లోకి కాపీ చేసాం. తర్వాత second.dat అనే ఫైల్‌లో వివరాలను ఒక్కొక్కటిగా ప్రింట్ చేశాం.

ఇప్పటి వరకూ మనం printf అనే ప్రమేయాన్ని, తెరపై విలువలను ప్రింట్ చేయటానికి ఉపయోగించాం. అదే fprintf అనే ప్రమేయాన్ని ఉపయోగించి ఏదైనా ఫైల్‌లోకి వివరాలను ప్రింట్ చేయవచ్చు. అదే scanf ను కీబోర్డు నుంచి వివరాలను తీసుకోడానికి ఉపయోగించాం. అదే fscanf ను ఏదైనా file నుంచి వివరాలను తీసుకోడానికి ఉపయోగించవచ్చు. fprintf, fscanf లను ఏవిధంగా ఉపయోగించాలో కింది ప్రోగ్రాంలో చూద్దాం. ఒక ఫైల్‌పై read, write రెండు ప్రక్రియలూ జరపాలంటే ఆఫైల్‌ను append mode లో ఓపెన్ చేయాలి. ఇప్పుడు Hello.dat ఫైల్‌ను append mode లో ఓపెన్ చేసి చూద్దాం.

```
#include<stdio.h>
#include<conio.h>
main()
{
    char c;
    FILE *fopen();
        *fp;
    fp=fopen("Hello.dat", "a");
    fprintf(fp, "\n Here is the new matter \n");
    fclose(fp);
    fp=fopen("Hello.dat", "r");
```

```
    printf("The file now contains \n");
    while(c=getc(fp)!=EOF)
        putchar(c);
        fclose(fp);
}
```

ఈ ప్రోగ్రాం రన్ చేస్తే

```
The file now contains
Good morning
How are you?
Here is the new matter
```

ఈ విధంగా ఫైళ్లని సృష్టించి అందులోకి వివరాలను ఇవ్వటం, తిరిగి గ్రహించటం, కొత్త వివరాలను ఇవ్వటం ంటి ప్రక్రియలను చేయవచ్చు.

```
#include<stdio.h>
main()
{
    int i;
    char *ch, word[20];
    void send-word(char * c);
    printf("Enter a word \n");
    scanf("%s", word);
}
void send_word(char * c)
{
    printf("The word in main is %s", c);
}
```

ఇప్పుడు ప్రోగ్రాంను ఎగ్జిక్యూట్ చేస్తే పైన మనం ఇచ్చిన పదం కింద ప్రింటవుతుంది. ఇంతకు ముందు మనం ప్రమేయంలోకి ఒక చరరాశిద్వారా ఒకే విలువను పంపటం చూశాం. అయితే ఇప్పుడు పదం మొత్తం ఎలావచ్చింది?

ఇక్కడకూడా ఒకటే విలువను పంపాం. శ్రేణిలోని మొదటి చరరాశిచిరునామా విలువను పంపాం. దాన్ని బట్టి శ్రేణిలోకి మిగిలిన విలువలను (అక్షరాలను) ప్రోగ్రాం తీసుకుని ప్రింట్ చేస్తుంది.

✦✦✦

అధ్యాయం 13

యునిక్స్ (UNIX)

'సి' కంపైలర్ యునిక్స్‌లో కూడా ఉంటుంది. యునిక్స్ నిజానికి చాలా శక్తిమంతమైన ఆపరేటింగ్ సిస్టమ్. డాస్‌లో రాసిన 'సి' ప్రోగ్రామ్‌లనే యునిక్స్‌లో కూడా చేయవచ్చు. అయితే కొన్ని భేదాలున్నాయి. యునిక్స్‌లో కొన్ని ఆదేశాలను, కంపైలర్‌ను ఈ పాఠంలో చూద్దాం.

యునిక్స్‌లో కూడా డాస్‌లో లాగా కమాండ్స్ ఉంటాయి. అయితే డాస్‌లో C:\ అని వస్తే యునిక్స్‌లో $ ప్రాంప్ట్ ఉంటుంది. ఈ ఆపరేటింగ్ సిస్టమ్‌లో కొన్ని ఆదేశాలను ఇప్పుడు పరిశీలిద్దాం.

ls - ప్రస్తుత డైరెక్టరీలో ఉన్న ఫైళ్లను లిస్ట్ చేయడానికి ఈ ఆదేశాన్ని ఉపయోగిస్తారు. డాస్‌లో DIR వంటిదే ఈ ఆదేశం. ls పక్కన డైరెక్టరీ పేరును ఇస్తే ఆ డైరెక్టరీలో ఫైళ్లు ప్రత్యక్షమవుతాయి.

cp:- ఒక ఫైల్‌లోని వివరాలను మరో ఫైల్‌లోకి కాపీ చేయడానికి ఈ ఆదేశాన్ని ఉపయోగిస్తారు.

cd- ఒక డైరెక్టరీ నుంచి మరో డైరెక్టరీకి మారటానికి ఈ ఆదేశాన్ని ఉపయోగిస్తారు. DOS లో cd ఆదేశం వంటిదే ఇదికూడా.

cat- ఫైల్‌లో ఉన్న వివరాలను తెరపై చూపటానికి ఈ ఆదేశాన్ని ఉపయోగిస్తారు. DOS లో TYPE వంటిదే ఈ ఆదేశం కూడా. cat పక్కన ఒకటికాని అంతకన్న ఎక్కువ ఫైల్ పేర్లను ఉంచవచ్చు.

cat file_1 file_2.....

mv: ఒక ఫైల్‌ను మరో డైరెక్టరీలోని మార్చటానికి ఈ ఆదేశం ఉపయోగపడుతుంది. డాస్‌లో move వంటి ఈ ఆదేశం ఫైల్ పేరు మార్చడానికి కూడా వాడవచ్చు.

man: యునిక్స్‌లో ఆదేశాలు లేదా పదాల గురించి వివరాలు తెలియాలంటే, man ఆదేశాన్ని ఉపయోగించవచ్చు. man పక్కన మనకు కావలసిన ఆదేశం లేదా పదాన్ని రాస్తే దాని గురించిన వివరణ ప్రత్యక్షమవుతుంది.

mk dir : ఒక కొత్త డైరెక్టరీని సృష్టించడానికి దాని ఉపయోగించవచ్చు.

యునిక్స్‌లో 'సి' ప్రోగ్రామ్‌లు రాసే విధానం :

యునిక్స్‌లో ఎడిటర్‌లు ఉంటాయి. వాటిలో ఒకదానిలో 'సి' ప్రోగ్రాంను రాసి సేవ్ చేసుకుని బయటికి వచ్చి కంపైల్ చేయాలి. తర్వాత అన్న ఎగ్జిక్యూటబుల్ ఫైల్ వస్తుంది. దాన్ని ఎగ్జిక్యూట్ చేస్తే ఫలితం వస్తుంది.

* * *

ముందుగా EMACS అన్న ఎడిటర్‌గా తీసుకుందాం.

emacs hello.c

అన్న ఆదేశం ఇస్తే hello.c అన్న ఫైల్ సృష్టించబడుతుంది.

ఆ ఫైల్‌ను కంపైల్ చేయాలి.

cc hello.c

అనే ఆదేశాన్ని ఇస్తే ఫైల్ కంపైల్ అవుతుంది.

ప్రోగ్రాంలో తప్పులు ఏవైనా వస్తే మళ్లీ ఎడిటర్‌ను ఓపెన్ చేసి తప్పులను సరిచేయాలి.

మళ్లీ ప్రోగ్రాంను కంపైల్ చేయాలి. ప్రోగ్రాం తప్పులు లేకుండా ఉంటే ఎటువంటి సూచన లేకుండా క్లోజ్ అవుతుంది.

అప్పుడు

a.out

అన్న ఆదేశం ఇస్తే ఫైల్ ఎగ్జిక్యూట్ అవుతుంది.

యునిక్స్‌లో ఎగ్జిక్యుటబుల్ ఫైల్ ఇది. ఏ 'సి' ప్రోగ్రాం కంపైల్ అయిన ఎగ్జిక్యూటబుల్ ఫైల్ మాత్రం ఇదే వస్తుంది.

మనం దాని పేరును మార్చటం ద్వారా మన ఫైల్ రూపంలోకి మార్చుకోవచ్చు.

ఉదాహరణకు

mv a.out hello

అన్న ఆదేశంతో hello అనేది ఎగ్జిక్యూటబుల్ ఫైల్‌గా మార్చవచ్చు.

◆◆◆

ఆఖరి బంగారు పలుకులు

ఈ పుస్తకం పూర్తిగా చదివారు కదా! మీ అభిప్రాయం ఏమిటి?

రామాయణంలో ఒక సామెత వుంది. "మంచి అనేది ఎక్కడవున్నా వెంటనే పెంచు", చెడు ఎక్కడ వున్నా తొలగించు లేదా తగ్గించు" అన్న అద్భుత సామెతను అనుసరిస్తే లోకం నిత్యకళ్యాణం అవుతుంది.

గత ఆరేడు సంవత్సరాలుగా కంప్యూటర్ విద్య గురించి పాఠకులలో అవగాహన కల్పించడంకోసం మాతృభాషలో మేము ప్రచురిస్తున్న కంప్యూటర్ పుస్తకాలు అత్యంత ఆదరణ పొందిన విషయం మీకు తెలిసిందే! ఈ పుస్తకం మీకు నచ్చితే వెంటనే ఇద్దరు ముగ్గురికి చెప్పండి. నచ్చకపోతే 'దండుగ! ఈ పుస్తకానికి డబ్బు, సమయం వృధా' అని చెప్పండి. అందువల్ల మంచి రెట్టింపు అవుతుంది. చెడు తగ్గిపోతుంది లేదా తొలిగిపోతుంది.

మాతృభాషలో కంప్యూటర్ నేర్చుకోవాలన్న పాఠకుడి కలని నిజం చేయడమే. ఈ పుస్తకం ఉద్దేశం. ఇప్పటికీ సరైన కంప్యూటర్ విద్యకు నోచుకోక శిక్షణ కోసం ఏ ఇనిస్టిట్యూట్‌ని నమ్మాలో తెలియక అయోమయంలో చాలా మంది పాఠకులు కొట్టుమిట్టాడుతున్నారు. అసలు కంప్యూటర్ ఇన్‌స్టిట్యూట్‌ల అవసరమే లేకుండా ఇంట్లోనే కూర్చోని స్వంతంగా నేర్చుకునేట్లుగా రూపొందింపబడ్డాయి మా పుస్తకాలు. అందుకే ఈ పుస్తకం నచ్చితే మరి కొంత మందికి చెప్పండి. ఇది మీరు మాకిచ్చే గురుదక్షిణ అనుకోండి, లేదా సమాజ సంక్షేమం కోసమని భావించండి. అలాగే పుస్తకాలు నచ్చకపోతే నిర్మొహమాటంగా ఆ విషయం పదిమందికి చెప్పి, వారి డబ్బు, సమయాన్ని కాపాడండి.

తమసోమ జ్యోతిర్గమయ !

మీ

పబ్లిషర్